1989/90 Data Book

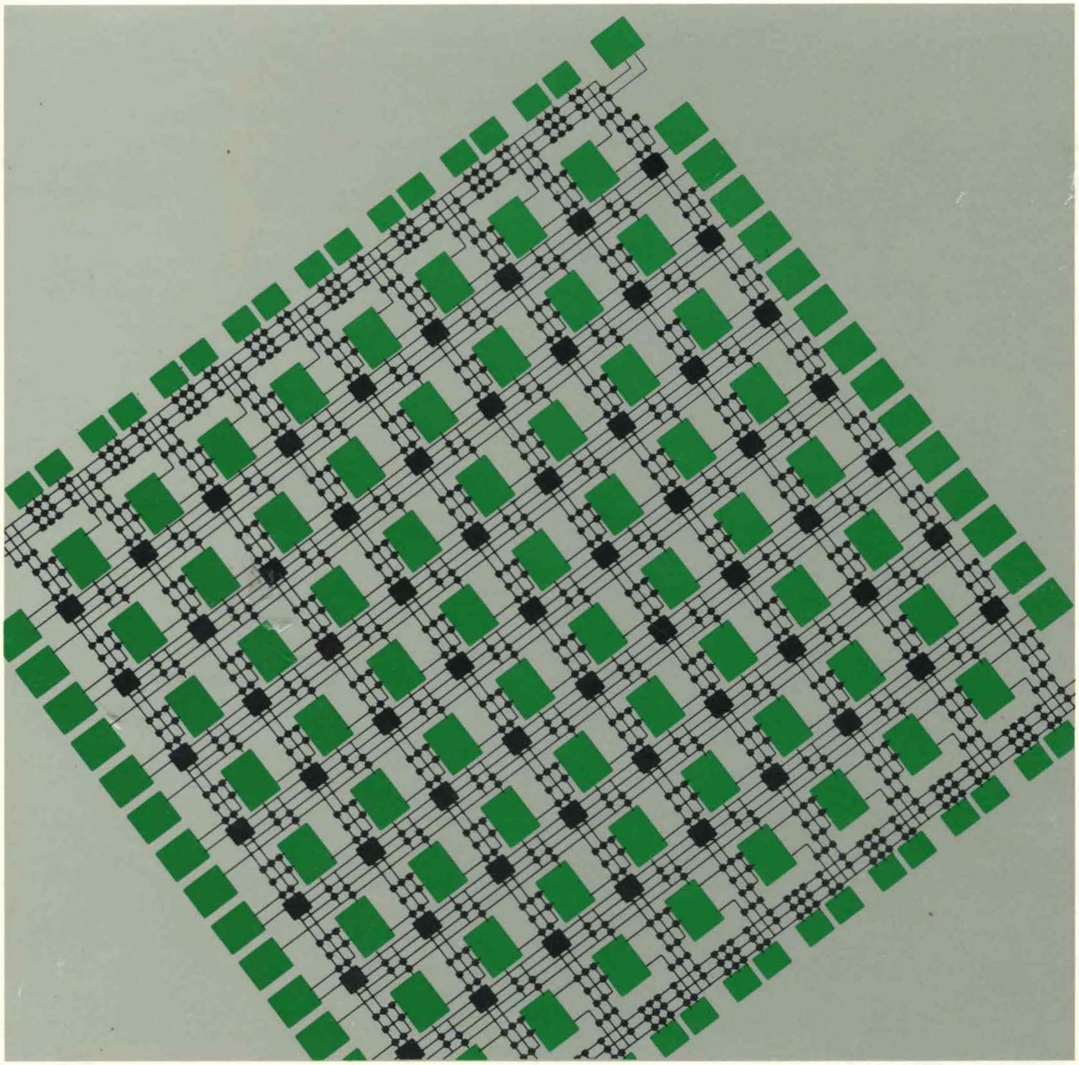
Programmable Gate Arrays

AMD

Programmable Gate Arrays

1989/1990 Data Book

Advanced
Micro
Devices



Advanced
Micro
Devices

PGA Databook

This book contains information about AMD's Programmable Gate Arrays, an exciting extension of our commitment to the field of programmable logic. The leader in programmable logic products, AMD continues to provide you with the quality, reliability and innovation you demand. As with every product we sell, AMD's Programmable Gate Arrays are backed by an extensive force of knowledgeable sales personnel and fully trained field applications engineers. After reviewing the information in this book, you will see how PGAs can fit into your applications. Please contact your local AMD sales office, authorized representative or franchised distributor so that we can, together, solve your technical problems with AMD's Programmable Gate Arrays.

Michael J. Callahan
Senior Vice President
Programmable Products Group

Table of Contents

Trademarks

Introduction 1

Applications 2

Product Information 3

Table of Contents

Advanced
Micro
Devices

AN-182

Configuring the LCA Device

By Ed Valleau

2

Logic Cell™ Array is a trademark of XILINX Inc.

Configuring the LCA Device

By Ed Valleau

Introduction

The Logic Cell™ Array device from Monolithic Memories is a new revolution in programmable logic. It offers the user gate array logic density while still providing the versatility of a programmable logic device (PLD). Its architecture is based on an array of static RAM cells, which must be configured when power is initially applied to the circuit. This gives the LCA device a high degree of flexibility since the functional logic contained in the device may be changed during development, thus shortening prototyping cycles. To update existing equipment, logic circuits may be modified with new configuration data. It is also possible to reconfigure the LCA device "on the fly" without the need for hardware changes. This would prove specifically useful in systems that need to change their protocols of operation dynamically.

Configuration data is stored in a nonvolatile source and read by the LCA device at "power up". The time it takes to perform a configuration is variable, from 12 to 34 ms depending on device type and mode of configuration. A typical storage medium for the LCA's configuration data would be an EPROM or EEPROM, which would be connected to the the LCA and accessed immediately after "power up". When the configuration process is complete and the device is fully functional, the EPROM or EEPROM may be put into a three-state condition and left deselected until the next configuration cycle.

There are a number of different modes of LCA device configuration, each mode being better suited to a particular operating environment or application than another mode. The following information outlines these various modes.

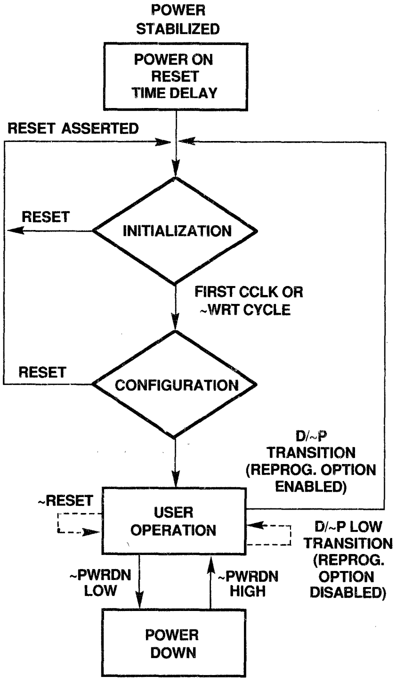


Figure 1. Configuration Sequence

Overview of the Configuration Process

The LCA device is comprised of three distinct programmable elements: Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs) and Programmable Interconnect. Because the device architecture is based on static RAM cells, the configuration of the CLBs, IOBs and Programmable Interconnect is random and indeterminate when power is first applied to the device. The LCA device goes through two states prior to being functionally operational. The first is a general initialization state, followed by a configuration sequence. (See Figure 1.)

The device uses a number of dedicated input/output pins to control the loading process. Configuration sequentially programs the RAM cells, ultimately creating functionally operational and interconnected logic blocks. The entire procedure is comparable to programming a PLD except that a conventional PLD is nonvolatile, and maintains its logic functionality when power is removed from the circuit.

The smaller of the two LCA devices, the M2064, is a 64-CLB device which requires exactly 12,040 binary bits of information to complete the configuration process. The M2018 is a 100-CLB device which needs 17,880 bits of information.

Table 1 shows the different modes that can be selected via the three dedicated mode input pins M0, M1 and M2. These will usually be hardwired to select a single mode. The LCA device reads the digital code applied to the mode pins prior to configuration, then enters one of five specific modes before becoming a functional logic system. The exact number of bits for successful configuration must be read into the LCA device, partial configuration is not possible.

CONFIGURATION MODE	SLAVE MODE	PERIPHERAL MODE	MASTER-HIGH MODE MASTER-LOW MODE	MASTER SERIAL MODE
Mode Selection code (M0:M1:M2)	1:1:1	1:0:1	0:0:1 (Master-Low) 0:1:1 (Master-High	0:0:0
Configuration data	Bit-serial	Bit-serial	Byte-parallel	Bit-serial
Automatic loading	No	No	Yes	Yes
Programming source	User Logic or Another LCA (Note 2)	CPU Data Bus Memory	External Byte-wide Memory	External Serial Memory
Number of user I/O pins required	2	6	25	3
Configuration time	Source Dependent (Note 1)	Source Dependent (Note 1)	12-24 ms (M2064) 17-34 ms (M2018) (Note 3)	12-24 ms (M2064) 17-34 ms (M2018) (Note 3)

Notes:

1. The minimum time in any case is approximately 12 ms for the M2064 and 17 ms for the M2018.
2. Also used by Monolithic Memories' XACTOR for In-Circuit Emulation.
3. This parameter depends on internal timing circuits and is manufacturing process-dependent. Therefore it may vary from device to device within the limits shown.

Table 1. Comparison of Configuration Modes

Choice of Configuration Mode

The choice of a configuration mode is influenced by the actual operating environment. For example, questions that might arise are: is the LCA being used as a standalone logic unit, or with a microprocessor? Can it be configured by a serial link? Other considerations include whether pins used for configuration are also used for functional operation, and whether or not these pins should be isolated from activating external logic during configuration. The designer has a choice of configuration modes for multiple LCA designs, and can use either parallel or serial support for a master mode LCA device which in turn can configure a slave or number of slave LCA devices.

The Configuration Pins

The pins used to configure the LCA device come in two forms: dedicated configuration pins which are used exclusively for configuring the LCA device, and multifunction configuration pins which can be used as general-purpose I/Os after configuration has been completed.

The Six Dedicated Configuration Pins

PIN	DESCRIPTION
M0,M1	Mode Select Pins
CCLK	Configuration Clock
~Reset	Master Reset
D/~P	Done/~Program
~PWRDN	Power Down

The Multi-Function Configuration Pins

PIN	DESCRIPTION	USAGE
M2	Mode Select	Present in all configuration modes
Din	Configuration Data In	
Dout	Configuration Data Out	
HDC	High During Configuration	
LDC	Low During Configuration	
~CS0,~CS1,CS2	Chip Selects	Peripheral Mode Only
~Wrt	Write Strobe	
~RCLK	Read Strobe	Master Mode Only
A0-A15	Address Lines	
D0-D7	Input Data Lines	

Pin Names and Functional Description

Done/Program Pin D/~P

The Done/~Program D/~P pin performs a dual function. It is an open-drain output with an internal programmable pull-up resistor. During configuration the pin is an output that is driven LOW by the LCA and can be monitored by external logic to determine whether or not the LCA is ready for functional use. When configuration is completed this pin is pulled HIGH by an internal programmable pull-up resistor of 3KΩ value. In a multi-LCA environment, the D/~P pin goes HIGH one clock cycle before the configuration is complete allowing time for user I/O signals to propagate between other LCA devices before entering the functional mode of operation. The LCA may be reconfigured at any time by pulling the D/~P input LOW with an external logic open-drain (open-collector) driver. When the internal logic recognizes a LOW input it drives the D/~P output LOW forcing a reconfiguration cycle.

There is a time delay of several microseconds before an active LOW input is recognized, so random triggering due to short noise bursts is highly unlikely. If the D/~P input is prevented from going HIGH immediately after configuration, then further reconfigurations will be suppressed and functional operation will not commence until that input is permitted to go HIGH. In multiple LCA designs the D/~P pins are wire-OR 'ed, so the last LCA device to be configured will prevent other, already configured devices, from functioning. When the last device is configured the composite wire-OR line goes HIGH and all LCA devices enter a functional mode simultaneously.

~RESET

The ~RESET input is active LOW and is used to start the initialization process before or during configuration, but not during functional operation. Figure 1 shows the initialization and configuration procedure as a flow diagram. Asserting the ~RESET line will put the LCA device into an initialization mode, aborting the current cycle of configuration.

Once configuration is complete the ~RESET input takes on a different role. It may be used to reset the device as if the device were a functional logic system. When asserted, internal registers and/or latches are reset. The reset condition may then be relaxed and functional operation resumed without the generation of a reconfiguration cycle.

To initiate reconfiguration the D/~P pin should be used instead of the ~RESET input. When ~RESET is LOW at the initialization cycle, its LOW-to-HIGH transition will sample the logic condition at the mode input pins The logic state sampled determines the mode of configuration. Once configuration has started these pins are not required to be held. However, it is recommended that these pins remain unchanged until the configuration phase is completed.

MODE CONFIGURATION INPUTS M0, M1, and M2

M0, M1 and M2 are the Configuration Mode Select Pins. They force the LCA device into the selected configuration mode on the rising edge of ~RESET. Pins M0 and M1 are dedicated mode inputs having no general-purpose I/O capability. They will usually be hardwired to logic HIGH or LOW conditions. For all modes except the master serial mode, M2 is HIGH and is pulled HIGH internally during configuration. After configuration this pin can be used for general I/O functions. M2 and M0 both have internal pull-up resistors. M1 should be tied to VCC or GND. Table 2 lists five of the modes of operation. The only mode that requires a logic LOW on the input to M2 is the Master Serial Mode which is used with a dedicated serial port EPROM.

M0	M1	M2	MODE SELECT
0	0	0	Master Serial Mode
0	0	1	Master LOW Mode
0	1	1	Master HIGH Mode
1	0	1	Peripheral Mode
1	1	1	Slave Mode

Table 2.

PWRDWN~

The PWRDWN~ input is an active LOW power-down pin that can be used to reduce power consumption of the LCA device when it is not being used. The LCA device is then considered off-line because all I/O lines are disabled while internal configuration is maintained. The VCC input may be reduced to 2.0 Volts and configuration data will not be lost. With internal logic disabled and I/O pins in a high impedance condition, the D/~P pin is forced LOW and all internal storage elements become cleared. It is essential that this pin is used only while D/~P is HIGH after initialization and configuration. It is common to tie this input to a valid logic HIGH if the power-down feature is not required.

Non-Dedicated Pins used in all Configuration Modes

DIN and DOUT

DIN (configuration Data IN) is used as a serial data port into the LCA device. Individual data bits are applied to this input and clocked into the device by rising clock edges applied to the CCLK input. DIN is used in both slave and peripheral modes.

DOUT (configuration Data OUT) is used to configure multiple LCA devices in a daisy chain. The DOUT signal from the preceding LCA device drives the DIN of the succeeding device and is synchronized to rising edge of the clock pulse on the CCLK line.

CCLK

CCLK (Configuration CLocK) as described above is used to synchronize the serial data stream into the data input DIN pin. The CCLK can be an input pin or output pin depending on the mode of configuration. It is an output for every mode of configuration except for the slave mode. The CCLK reverts to an input in the slave mode, and can be driven either by an external source or the CCLK output of another LCA device configured in the master mode. In the operational mode the CCLK is an input to enable configuration data to be read back from the LCA via the M0 and M1 pins, which have the dual function of Readback Trigger (RT) and Readback Data (RD), respectively. CCLK has an internal pull-up resistor which allows the input to be pulled HIGH when not in use.

RCLK

The Read CLocK is used as an output signal which goes LOW when the LCA device expects to see valid input data, and HIGH when the address bus contents are changing states. It can be used to enable the external EPROM, and in the master serial mode can be used to clock an external address counter. This pin is used to clock the CLK input of the serial EPROM, a support device for the LCA.

HDC and LDC

High During Configuration (HDC) and Low During Configuration (LDC) are driven HIGH and LOW, respectively, for the duration of the configuration process. They are used to control external logic during configuration. When configuration has been completed HDC and LDC become general-purpose input/output pins. All of the other I/O pins not involved in the actual configuration process are connected to internal pull-ups to VCC, which are removed after configuration.

CHIP SELECT INPUTS ~CS0 ~CS1 CS2 ~WRT

~CS0, ~CS1, and CS2 are the chip select pins used during the Peripheral Configuration Mode only. The three enables can be used to map the LCA device to a specific address. Active LOW inputs to ~CS1 and ~CS0, and an active HIGH input to CS2 will select the LCA device as if it were a memory or peripheral location mapped into the address space of a microprocessor system.

~WRT in the Peripheral Configuration Mode is the same as CCLK in the slave mode. A single data bit is transferred from the data bus into the LCA device on each rising edge of ~WRT.

A0-A15 and D0-D7

The address and data pins are used in the master parallel modes only, and are converted to general-purpose input/output pins after successful configuration. A0-A15 are the address lines used to access the external EPROM. In the Master Low mode, address location 0000 hexadecimal is read first, and the addresses increment each time a data byte is read until all of the configuration data is loaded into the LCA device. The D/~P then goes HIGH indicating that the device is loaded. The address counter outputs convert to general-purpose I/O, unless the D/~P pin is held LOW by the wire-OR function of a D/~P output from a slave device. In that case the counters will continue to count, downloading the slave's configuration data. Counting will continue for more than one slave. In the Master High Mode, address FFFF hexadecimal is accessed first, and the address lines are decremented. This allows the LCA device to share addressing space with an EPROM or EEPROM used by the system. D0-D7 are the data lines connected to the external memory device.

Slave Configuration <M0 M1 M2> = <1 1 1>

The Slave Configuration Mode (Figure 2a) is simple to implement requiring only three pins, two lines to perform handshake and synchronization and one line for data transmission. Slave configuration is used in the XACT Development System to download data to the support hardware. In this mode data is presented to the LCA device as a serial bit stream, which is transferred to the device as if it were a very large shift register. The data is presented to the DIN pin and sequentially clocked into the device using the CCLK input. When the device is completely loaded, the D/~P pin goes HIGH to confirm that configuration is complete and that the device is ready to function as a programmed unit.

In Figure 2a the LCA device is shown connected to a microcomputer/microprocessor port with the data line (D0) connected to the DIN input. The rising edge of the strobe output will clock valid data on D0 into the LCA. The process continues until the D/~P confirms that configuration is complete to the microcomputer via the data input (D7). The microcomputer can poll D7 to

PIN NAME	APPLICABLE CONFIGURATION MODE(S)					FUNCTION DURING CONFIGURATION	FUNCTION DURING USER OPERATION
	S	P	MH	ML	MS		
M0	•	•	•	•	•	Mode select 0 (I)	Readback trigger (I)
M1	•	•	•	•	•	Mode select 1 (I)	Readback data out (O)
M2	•	•	•	•	•	Mode select 2 (I)	<User I/O>
D/P (Note 1)	•	•	•	•	•	Indicates when configuration process is done (O)	Initiates/Inhibits Reconfiguration (I)
RESET (Note 1)	•	•	•	•	•	Abort/Restart configuration (I)	Master clear of all internal Flip-Flops (I)
CCLK	•	•	•	•	—	Configuration clock (See Notes 1 and 2)	Readback clock (I)
DIN	•	•	—	—	•	Configuration data in (I)	<User I/O> (Note 3)
DOUT	•	•	•	•	•	Configuration data out (O)	<User I/O>
HDC	•	•	•	•	•	Logic HIGH (O)	<User I/O>
LDC	•	•	•	•	•	Logic LOW (O)	<User I/O>
A0-A15	—	—	•	•	—	Address bus (O)	<User I/O>
D0-D7	—	—	•	•	—	Data bus (I)	<User I/O> (Note 3)
RCLK	—	—	•	•	•	Read clock (O)	<User I/O>
WRT	—	•	—	—	—	Write strobe (I)	<User I/O>
CS0	—	•	—	—	—	Chip select 0 (I)	<User I/O>
CS1	—	•	—	—	—	Chip select 1 (I)	<User I/O>
CS2	—	•	—	—	—	Chip select 2 (I)	<User I/O>

Notes:

1. The RESET, CCLK, and D/P pins have multiple functions. See text for further details.
2. During Slave mode configuration, the CCLK pin is an input, while for all other modes, it is an output.
3. DIN and D0 are the same physical pins but are associated with different configuration modes.

Abbreviations:

S = Slave	MH = Master high
P = Peripheral	ML = Master low
I = Input	MS = Master serial
O = Output	

Table 3. Summary of Pins Used for Configuration

determine the configuration status, or set input D7 to generate an interrupt. Another application for the slave mode would be in a multiple LCA design where one LCA device can serially load subsequent slave LCA devices arranged in a daisy chain. Using LCA devices connected as master and slave or slaves is discussed more fully in a later section.

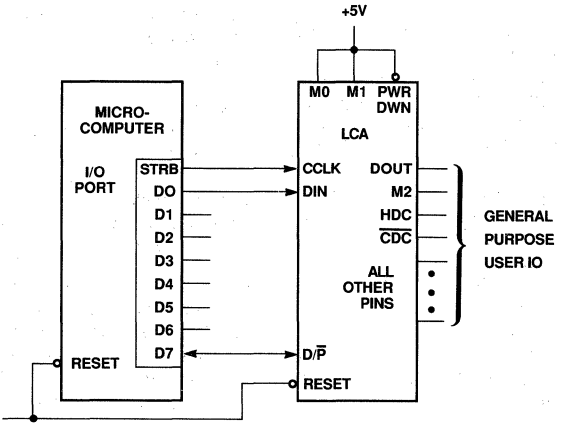


Figure 2a. Slave Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	HIGH	Mode Select
M1	25	17	Input	HIGH	Mode Select
CCLK	60	42	Input	<Clock>	Configuration Clock
RESET	44	31	Input	HIGH	Master Reset
D/$\overline{P}$	45	32	Output	LOW	Done/Program
PWRDWN	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
DOUT	59	41	Output	<Data>	Configuration Data Out
HDC	28	20	Output	HIGH	Constant "1" Level
LDC	30	21	Output	LOW	Constant "0" Level

Figure 2b. Slave Mode Pin Summary

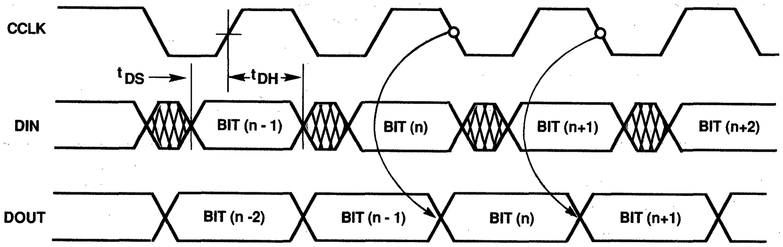


Figure 2c. Slave Mode Configuration Timing

Peripheral Mode Configuration <M0 M1 M2> = <1 0 1>

The Peripheral Mode (Figure 3a) is similar to the Slave Mode in that the data is presented in a bit serial form. In the Peripheral mode, however the LCA device is configured as a microprocessor-compatible peripheral device. The microprocessor can access the LCA device directly through its internal address bus, and map it through chip select logic. Control of the write operation is direct to the LCA device. Less hardware is required to connect the LCA to a microprocessor in this mode, and the loading of data is serial over a single data line. Synchronization is achieved by the falling edge of a ~WRT input, while data is set up on the DIN line and strobed by the falling edge of a microprocessor-generated write signal.

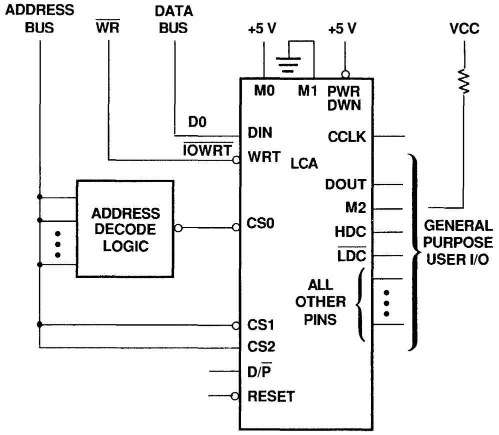


Figure 3a. Peripheral Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	HIGH	Mode Select
M1	25	17	Input	LOW	Mode Select
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
$\overline{\text{PWRDWN}}$	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
DOUT	59	41	Output	<Data>	Configuration Data Out
$\overline{\text{CS0}}$	50	35	Input	LOW	Chip select (Active LOW)
$\overline{\text{CS1}}$	51	36	Input	LOW	Chip select (Active LOW)
CS2	54	37	Input	HIGH	Chip select
$\overline{\text{WRT}}$	56	38	Input	<Strobed>	Write enable (Active LOW)
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level

Figure 3b. Peripheral Mode Pin Summary

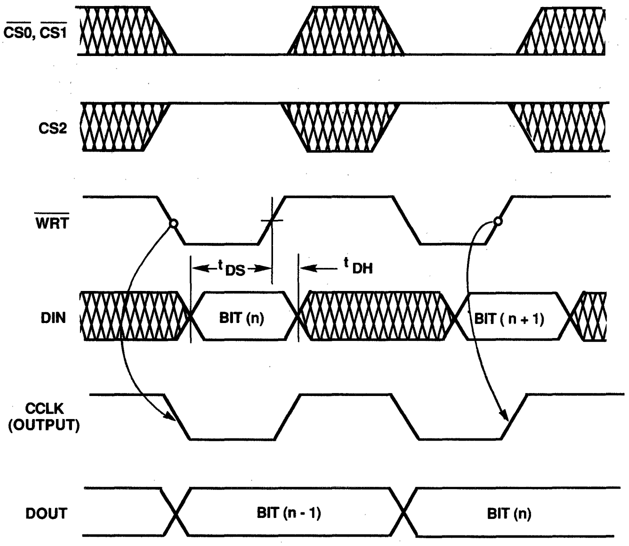


Figure 3c. Peripheral Mode Configuration Timing

Parallel Master Mode Configuration

MASTER MODE LOW <M0 M1 M2> = <0 0 1>

MASTER MODE HIGH <M0 M1 M2> = <0 1 1>

In the Master Low Parallel mode (Figure 4a) data is loaded in parallel at a rate determined by an on-chip timer in the LCA device. Configuration data is stored in an EPROM or EEPROM, and a parallel data path is connected from the memory output data lines D0-D7 to dedicated data inputs on the LCA device. There are sixteen address outputs, A0(LSB)-A15(MSB), connected to the EPROM. At the start of configuration in master mode low the LCA device sends an incrementing address starting at address 0000 hexadecimal and sequentially selects the configuration bytes. Data is serialized when loaded into the LCA device.

When configuration is complete the address and data lines become general-purpose I/O blocks. The D/~P pin is used to provide external logic, an indication that the LCA is currently functional or configuring.

The D/~P pin can be used to select the EPROM/EEPROM during configuration. An active LOW output applied to the memory's Chip Select input will select the device to read the configuration data and deselect it when configuration is complete. The method shown in Figure 4a is suggested for logic systems that have no host processor available to perform the configuration. A drawback is that address and data lines might require isolation from external logic circuits while the LCA device is configuring. The logic designer must determine which signals need isolation and which do not. Using IOBs which are not address and data lines might be a solution to avoid any conflict that might occur when external logic is driven during configuration. Also, the designer must avoid loading address and data lines with capacitive or inductive components. For example general IOBs should be used for circuits such as relaxation oscillators which require capacitive loads. Loading address or data lines with capacitors might lead to a violation of setup and hold times, causing erroneous configuration information to be read.

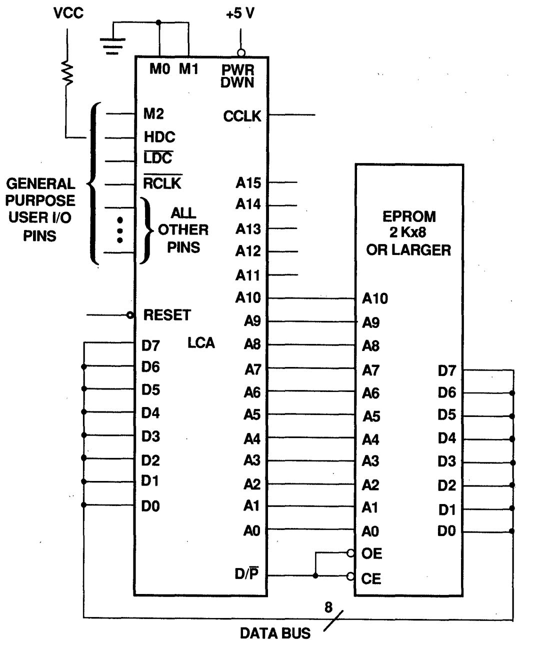


Figure 4a. Master Parallel Mode

In the Master Mode HIGH the initial starting address is hexadecimal FFFF, and the address decrements after each data byte is read. For both Master Mode HIGH and Master Mode LOW, the LCA device will set the D/~P pin HIGH after enough bits have been read to configure the one master LCA device. This permits configuration data to be stored in an EPROM that also holds microprocessor firmware. Only 1505 (5E1 hex) bytes are required for an M2064 LCA device and 2235 (8BB hex) bytes for the M2018, so large EPROMs can store configuration data for a number of LCA devices. Configuration information is concatenated in one EPROM and read by the master device, then sent to slave LCA devices as outlined in the section describing the configuring of multiple LCA devices.

Bytewide Master Mode Pin Summary

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	LOW	Mode Select
M1	25	17	Input	LOW or HIGH	(Master-low mode) (Master-high mode)
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
$\overline{\text{PWRDWN}}$	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DOUT	59	41	Output	<Data>	Configuration Data Out
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level
$\overline{\text{RCLK}}$	57	39	Output	<Strobed>	Chip enable output
A0-Axx	—	—	Outputs	<Address>	Memory address bus
				A15 A11	A0
	48 DIP			3 5 6 4 2 1	48 47 46 45 44 43
68 PLCC				65 67 2 4 6 8 9 7 5 3	68 66 64 63 62 61
D0-D7	—	—	Inputs	<Data>	Memory data bus
				D7	D0
	48 DIP			28 29	34 35 36 37 38 40
68 PLCC				41 42	48 50 51 54 56 58

Table 4b. Master Mode Pin Summary

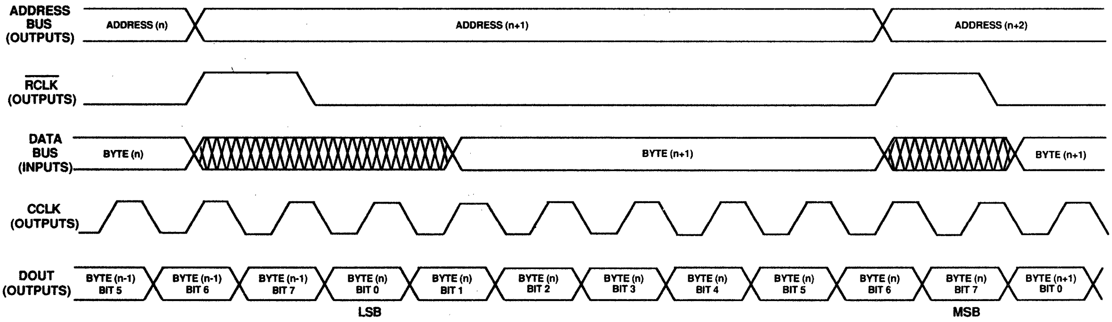


Figure 4c. Master Mode Configuration Timing

Serial Master Mode
<M0 M1 M2> = <0 0 0>

The Master Serial Mode (Figure 5) uses a Serial Data PROM (SDPROM) which has an internal address counter and a serial data port for the conversion of parallel data from the SDPROM to a single data output. Figure 5a shows the schematic diagram required for this mode. Configuration data is stored in the PROM section and downloaded to the LCA device in the most pin-efficient way, avoiding large numbers of address and data interconnections. The SDPROM is connected to the LCA device using only three pins. In this mode the LCA generates a clock signal which is used to increment the address counter built into the SDPROM. The SDPROM then sends data to the DIN pin on the LCA device. This mode is very similar to the Slave and Peripheral Modes in that data input is through the DIN pin. However, the SDPROM requires a clock input to increment its internal address counter which is supplied by the RCLK output of the LCA device. The configuration time is therefore determined by the LCA devices on chip counter. The D/~P input terminates the configuration process by disabling the SDPROM. One SDPROM can hold enough configuration data for three M2064 or two M2018 devices, and they can be cascaded for larger configuration arrays.

The data to be loaded into an LCA device is developed using the XACT Development Software, and is stored in one of two modes: a serial bit stream to be used in the Peripheral and Slave modes, or a 1500-byte PROM file for use in the Master Mode.

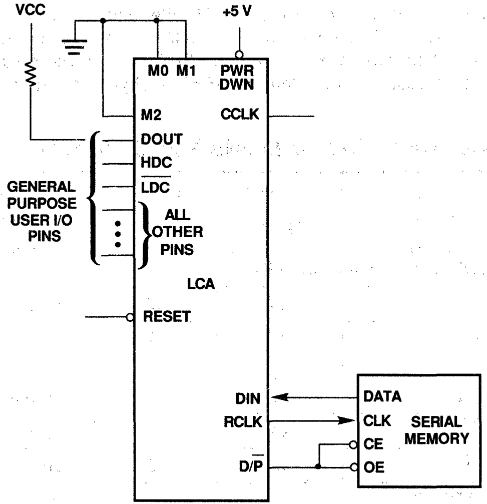


Figure 5a. Serial Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	LOW	Mode Select
M1	25	17	Input	LOW	Mode Select
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
PWRDWN	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	LOW	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
RCLK	57	39	Output	<Strobed>	SDPROM Clock
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level

Figure 5b. Master Serial Configuration Mode Pin Summary

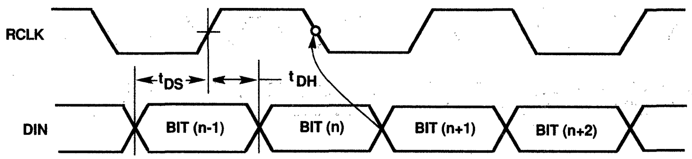


Figure 5c. Master Serial Mode Configuration Timing

Configuring Multiple LCAs

Recognizing that multiple LCAs could be used in a system, the designers added a feature which makes configuration easier by allowing several LCAs to be connected together in a daisy chain. In the Master mode, the first LCA reads data from the EPROM in parallel until it has received all of its configuration data. At this point its D/~P pin would normally be pulled passive HIGH but it is wire-ORed to the remaining LCA devices configured in the slave mode and is held LOW. The first LCA, although configured will not start to function, but continue to pass configuration data to the next LCA device. The data out line (DOUT) drives the DIN line of the first slave, and the CCLK input clocks the configuration data through until the next LCA is configured. The wire-OR action holds the D/~P input LOW preventing the configured LCAs from entering a functional mode. The sequence continues until the last LCA in the slave chain is configured and allows the D/~P line to go HIGH. All the configured devices in the chain start to function simultaneously because the composite wired-OR function goes HIGH on all LCAs simultaneously.

Due to the current sink capability of the D/~P input being able to handle only one pull-up load satisfactorily, only one of the LCA devices in the daisy chain should be configured with 'pull ups'. The first device in the chain can be configured in any of the four modes. All other devices in the chain would use the slave mode.

The daisy chain configuration mode is the easiest and most pin efficient, but the total configuration time increases linearly for each LCA added to the chain. It is also possible to configure LCAs in parallel in the slave or peripheral modes as shown in Figure 7. The total configuration time is then reduced to the configuration time of a single device.

Connecting parallel LCA's in the peripheral mode also allows the use of DMA transfer techniques to decrease configuration time.

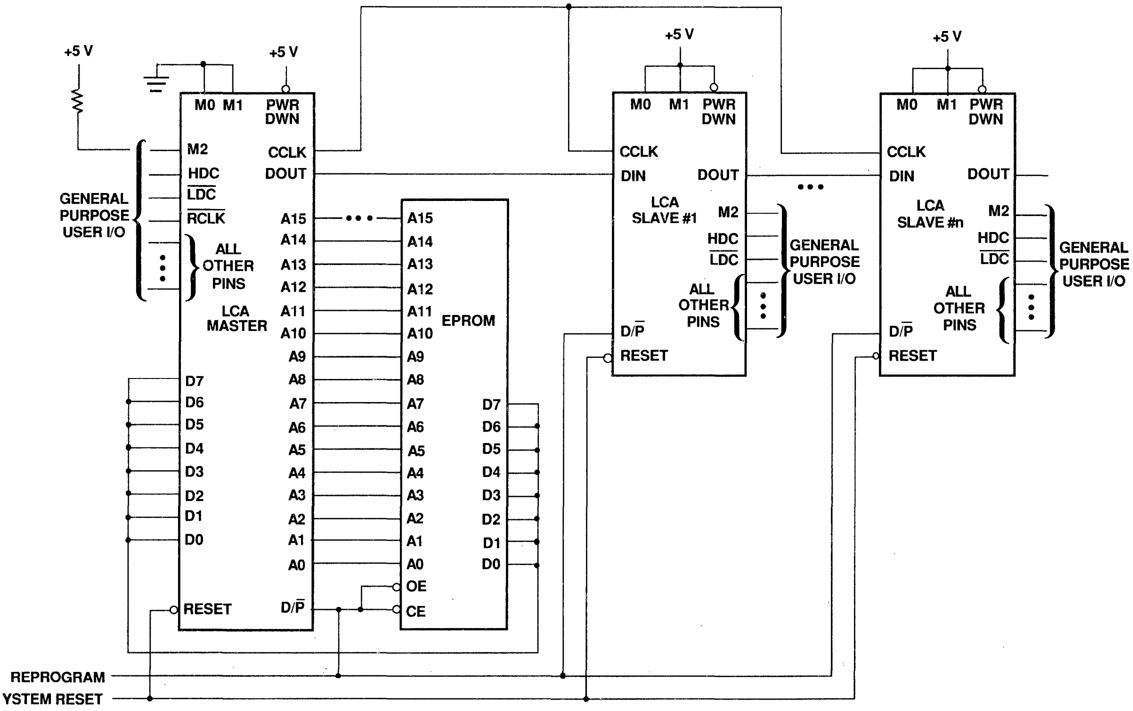


Figure 6a. Master Mode LCA with Daisy-Chain

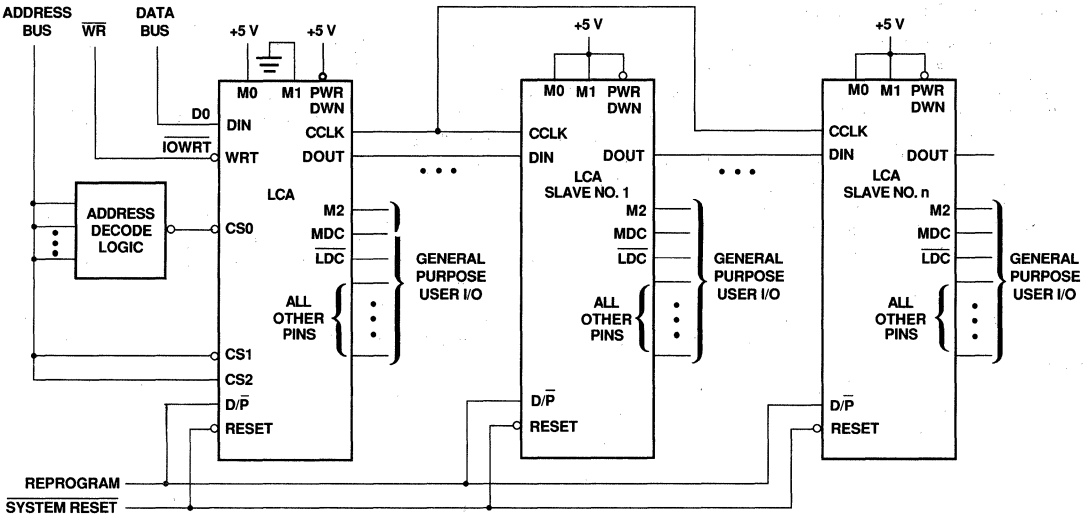


Figure 6b. Peripheral Mode LCA with Daisy-Chain

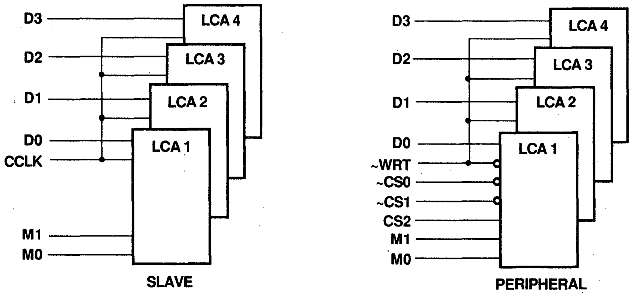
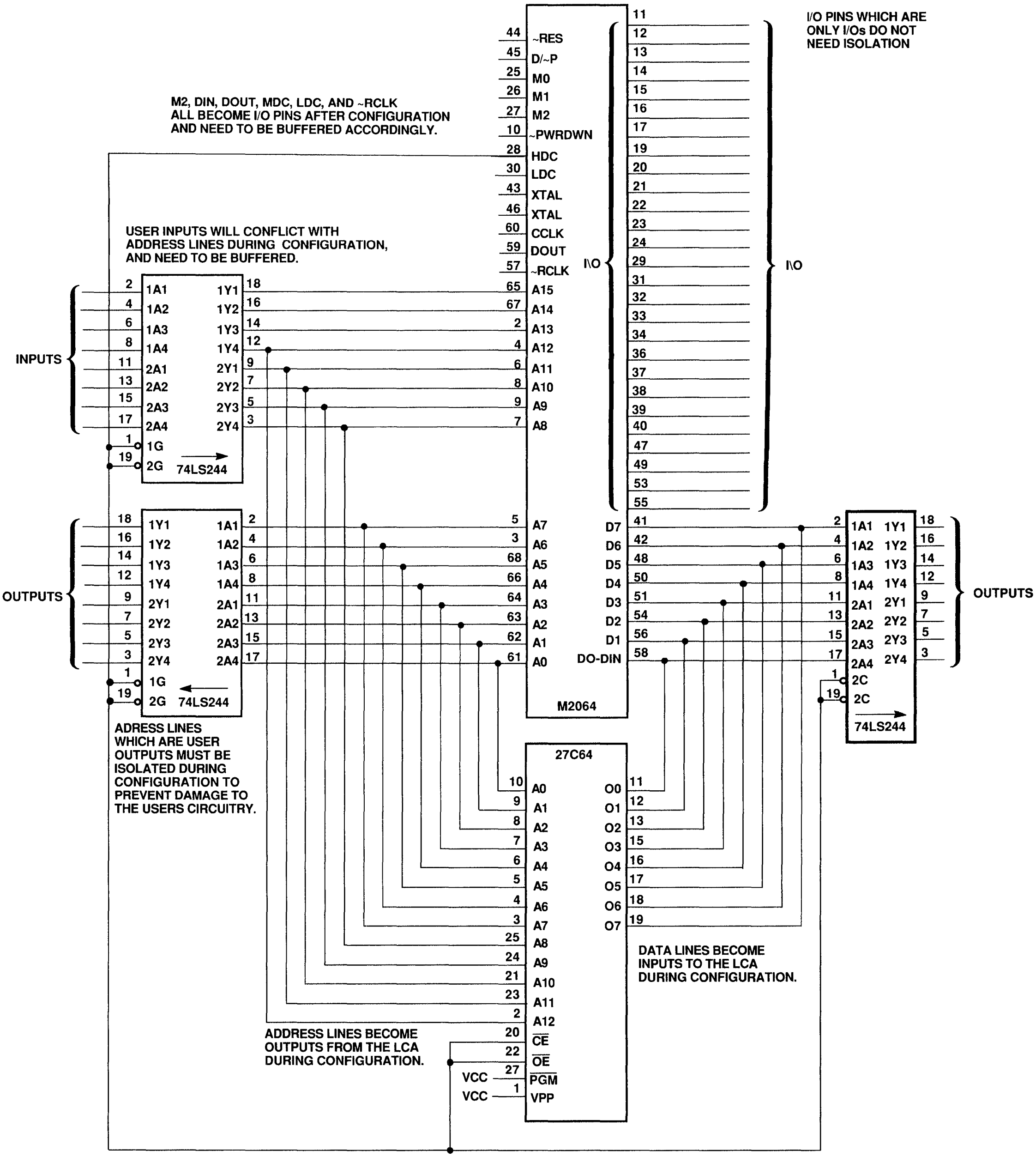


Figure 7. Parallel Loading of LCAs in Slave and Peripheral Modes

Use of the Multiple I/O Pins

When designing with the LCA, careful consideration must be given to which signals are connected to the multiple-function I/O pins. Conflicts can arise when a pin normally used as an input to the LCA suddenly becomes an output during the configuration cycle. Similarly, outputs which are normally under tight control by the LCA logic can transition erratically during configuration, causing adverse effects to the circuitry connected to them.

By simply assigning LCA outputs to pins which become outputs during configuration, and LCA inputs to pins which become inputs during configuration, the designer assures that conflicts can be held to a minimum. If all conflicts cannot be resolved using this method, then external buffers may be added to eliminate the the possibility of any bus contention. The signals HDC (High During Configuration) and LDC (Low During Configuration) can be used to enable or disable the buffers at the appropriate times. See Figure 8.



M2, DIN, DOUT, MDC, LDC, AND ~RCLK ALL BECOME I/O PINS AFTER CONFIGURATION AND NEED TO BE BUFFERED ACCORDINGLY.

I/O PINS WHICH ARE ONLY I/Os DO NOT NEED ISOLATION

USER INPUTS WILL CONFLICT WITH ADDRESS LINES DURING CONFIGURATION, AND NEED TO BE BUFFERED.

ADRESS LINES WHICH ARE USER OUTPUTS MUST BE ISOLATED DURING CONFIGURATION TO PREVENT DAMAGE TO THE USERS CIRCUITRY.

DATA LINES BECOME INPUTS TO THE LCA DURING CONFIGURATION.

ADDRESS LINES BECOME OUTPUTS FROM THE LCA DURING CONFIGURATION.

Figure 8. Isolation of I/O Pins During Master Mode Configuration

Advanced
Micro
Devices
AN-173

M2018 Provides Decoding for Six-Digit, Seven-Segment Liquid Crystal Display

Chris Jay

Abstract

The Logic Cell™ Array (LCA) from Monolithic Memories is a high density programmable device capable of supporting true VLSI logic functions. Its unique properties can be viewed as providing the benefits of Programmable Logic Devices (PLDs) while approaching the high functional density of a gate array. Combining these features makes the LCA device a product capable of bridging the gap between PLDs and Gate Arrays.

The LCA structure differs from the conventional concept of the PLD because it is based on a CMOS RAM cell architecture. Its internal logic circuits, input/output (I/O) resources plus the interconnect are all programmable. In comparison to the traditional fuse array of a PLD which is linked to a fixed logic structure. The efficiency of this logic structure reduces as functional density increases.

The low-power CMOS RAM locations of the LCA device must be initialized and configured immediately after power has been applied to the circuit. Once configured, functionality is maintained by the continued application of power to the device. To ease configuration, the device can reside alongside a low-cost EPROM which holds the configuration data. The LCA device can automatically load itself and be reprogrammed any number of times. This feature of reconfigurability can be used for system development, modifying existing designs, or supporting multiple system choice, with very little actual hardware modification. In many instances this can be achieved dynamically or "on the fly."

The following applications note describes the design of a six-digit, seven-segment decoder/driver for Liquid Crystal Displays (LCDs). Design methodology is outlined and the use of the XACT™ software is discussed in the role of a CAD design tool for the LCA device. The final designs are available to readers of this applications note on request.

M2018 Provides Decoding for Six-Digit, Seven-Segment Liquid Crystal Display

Chris Jay

Introduction

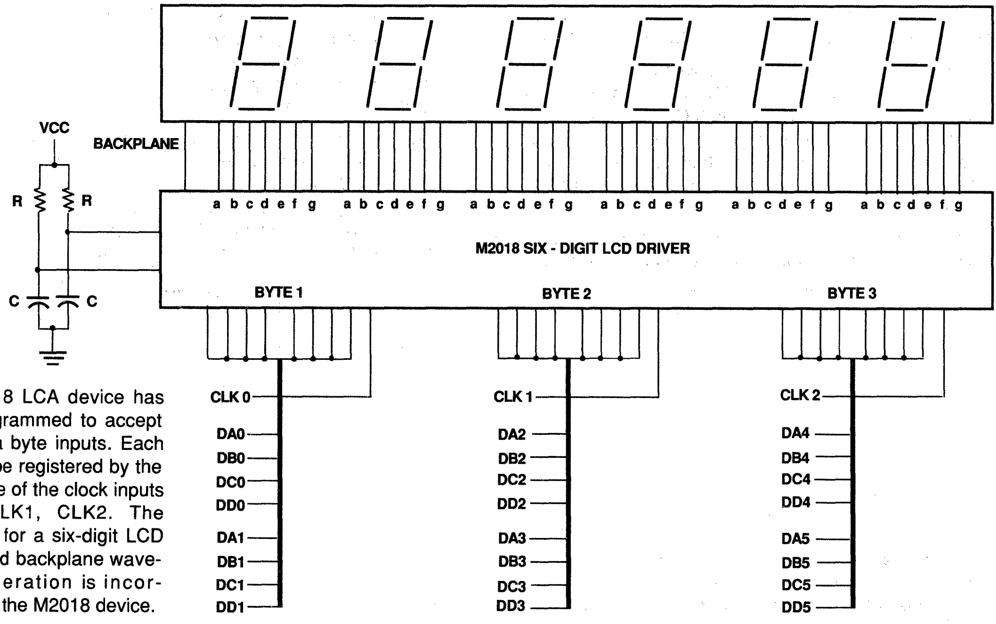
There are two types of low-voltage, seven-segment displays available as indicator panels of multimeters, frequency counters, tachometers and other digital instruments: the light emitting diode displays (LEDs), and Liquid Crystal Displays (LCDs). The advantages of the LCD are mainly associated with very low power consumption and ease of readability. Instruments and portable equipment can benefit from low-power CMOS technology and use these displays. The problem with LED displays is that a much higher operating current is required to illuminate the segments, virtually precluding their application in portable battery-operated equipment. Also, LED display outputs tend to "wash out" in sunlight, so they are very unsuitable in bright daylight, or where the ambient presence of light is high. In the display instrumentation of an automobile, for example, an LED display panel might be "washed out" by high levels of reflected sunlight. LCD displays are preferable because the visual quality is good even in brightly lit environments.

Despite the good visual quality and low-power consumption of the LCD display, an LED type can easily be driven from a multiplexed bus output. The output pin requirement of a seven-segment LED driver can be reduced considerably by a multiplexed output arrangement. In a six-digit display, each anode (or cathode) drive is selected in synchronism with its multiplexed seven-segment output. A total of thirteen pins are required, seven pins to drive the segments and six pins to control the anode (or cathode) of each display. If each digit is multiplexed at high speed with a proportionally larger pulse current per display, then the readability of the output is unaffected. With an LCD display, multiplexing is virtually impossible for two reasons. First, the LCD display has a slow response time and can not be multiplexed to give a good visual output. Second, the display backplane has to be pulsed. Any beating of a multiplexing frequency with the backplane bias frequency can occur, resulting in an unreadable output.

LCD displays have drawbacks because any driver circuit will need one pin per segment, plus a backplane driver pin for the whole display unit. Seven segments for each digit, in a six-digit display requires forty-two outputs pins, plus the one backplane pin. The input pin requirement is twenty-four, four pins for each of the six digits. The total pin count is sixty-seven. Additional input circuitry to support register enables and oscillator inputs would use up even more I/O resources and pins.

Block Function of the LCD Decoder/Driver

Figure 1 shows a schematic block diagram of the system as designed into the M2018 Logic Cell Array. Additional pins to synchronize the clocking of data into the internal data registers take three more pins. Since a backplane oscillator is required, two pins are configured to a resistor/capacitor (RC)



The M2018 LCA device has been programmed to accept three data byte inputs. Each byte can be registered by the rising edge of the clock inputs CLK0, CLK1, CLK2. The decoding for a six-digit LCD display and backplane waveform generation is incorporated in the M2018 device.

Figure 1. Liquid Crystal Display Decoder Driver

network. The RC combination generates a square wave which is used as a backplane bias to the display. The pin count is seventy-two.

Binary data applied to byte 1 is registered after the rising edge of CLK0. The registered data is decoded to illuminate the relevant segments forming the hexadecimal display. Decoded data is then routed to the two least significant digits of the six-digit LCD. Byte data applied to inputs 2 and 3 are loaded by a clock rising edge, CLK1 and CLK2, respectively. A backplane oscillator output drives the LCD's backplane with a low-frequency square wave, which is determined by the RC time constants of a resistor/capactitor network. To display a segment the decoder circuitry will drive the selected segment with a square wave that is 180 degrees phase-shifted from the backplane reference. If a segment is driven from an in-phase signal, then it is blanked.

The advantage of the M2018 Logic Cell Array in this application is that it is a CMOS device, and consumes low power at low operational frequencies. Its power consumption is dependant on the internal clocking frequency and also on the percent usage of on-chip logic. There are one hundred individually configured logic blocks in the M2018, and any logic blocks that remain unused will consume only static substrate leakage current. Since the M2018 contains seventy-four general-purpose user I/O blocks, there is ample programmable logic and I/O capability to perform decoding, registering and storage of data for six seven-segment digits.

LCA Configuration

Since the LCA device is based around a programmable array of volatile RAM cells, it must recieve configuration information at "power up." In this application it is programmed from an EPROM which holds the configuration data. Configuration takes place after power has been applied to the circuit, and it has been successfully reset. The LCA device enters a configuration mode, sequentially reads the data from the EPROM into its RAM cells, and becomes functionally operational in approximately twenty-five milliseconds. It disables the EPROM after configuration. The volatility of the M2018 is a distinct advantage in many applications. In this case, different display decode operations may be selected at "power up'. Higher order address lines may be "hardwired" to select different configuration patterns from separate pages in the EPROM. So a small amount of deferred design may be added to the system.

The M2018 has many advantages over custom VLSI circuits. Here, designers have control over how they wish to configure the circuit. Different display fonts can be programmed, for special characters. Most of the I/O pins can be reconfigured to ease printed circuit board layout.

Figure 2 shows the circuit connections of an 84-pin M2018 in a PLCC to an 8 Kbyte-wide EPROM. Approximately 2.5 Kbytes are required for configuration. Three configuration patterns may be cascaded into one 8 K x 8 EPROM, but in this application, two configuration patterns may be stored, one at base location of 0K and one at base location of 4 K. Address line A12 may be configured to a DIP switch or hardwire link, to select one of two configuration patterns. Two designs were developed, one for decoding binary data to a hexadecimal display and one for binary coded decimal. Both configuration patterns could be programmed into the EPROM and selected at a later time.

The configuration mode chosen for the LCA device was the Master Low mode. Data is sequentially read from the EPROM during the configuration cycle. After power has been applied and the RESET input has been deasserted, the M2018 will output a series of incrementing addresses to the memory starting at the initial address of 0000. The DONE/~PROG (D/~P), which is an output during configuration, is driven active LOW. It is tied to the CS and OE inputs of the EPROM. When configuration is finished, D/~P is pulled passive HIGH by an internally-configured "pull-up" resistor in the LCA device . The EPROM is then deselected. The normal data flow into the device is via dedicated input lines D0-D7 during configuration, and afterwards these become general-purpose I/O lines. The total time for configuration can vary between 17 to 34 ms.

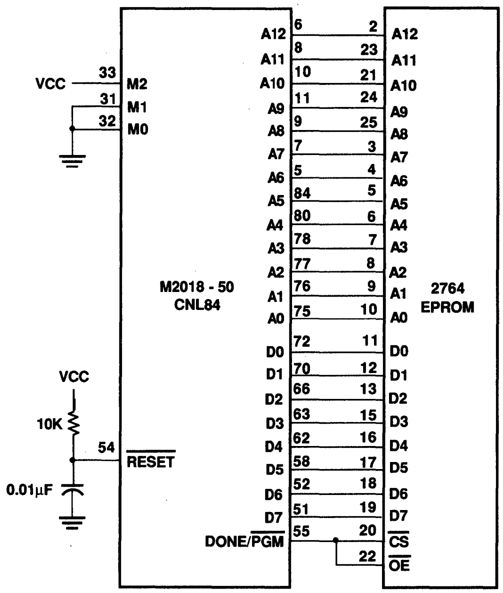


Figure 2. Liquid Crystal Display Decoder Driver

Design of the Backplane Oscillator

To establish a backplane bias frequency, an R/C network is used in conjunction with a logic block that is designed to function as a relaxation oscillator. The Configurable Logic Block (CLB) is connected to two Input Output Blocks (IOBs) which in turn are connected to two external R/C networks, R1,C1 and R2 and C2 as shown in Figure 3a. The calculation of the R/C values to create a low-frequency backplane oscillation of even mark space ratio is given below. For an even mark/space ratio R = R1 = R2 and C = C1 = C2. The time constants for period t are given by the following formula:

T1 = 0.35(C X R X 2) ;for TTL voltage ;thresholds.

T2 = 0.75(C X R X 2) ;for CMOS voltage ;thresholds.

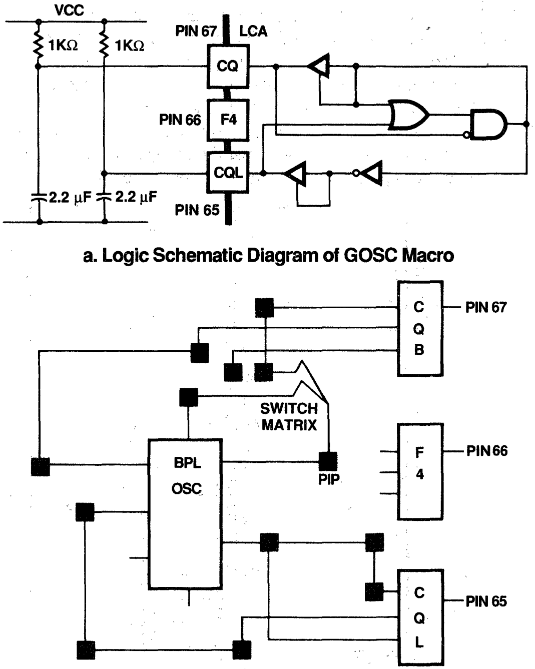
The general expression for the calculation becomes:

T = N X [(R1 X C1) + (R2 X C2)]......(1)

Where N is the TTL, or CMOS multiplying factor of 0.35 for TTL or 0.75 for CMOS. The LCA device can be programmed to be compatible with either technology.

The values of 1 Kohm and 2.2 microfarad gave a backplane oscillation frequency of about 80 Hz.

Figure 3a shows the schematic arrangement of the backplane oscillator as it would be configured in the LCA device. Two R/C networks of 1 Kohm and 2.2 microfarads are connected to pins 67 and 65 respectively. The placement of the blocks is shown in Figure 3b as it would be seen on the color monitor of a PC when the XACT Design Editor is used. Pins dedicated to configuration functions must not be loaded with components such as capacitors because they could prevent the LCA device from reading valid data. Pin 66, which is associated with data line D2 was therefore assigned to drive one of the LCD segments.



a. Logic Schematic Diagram of GOSC Macro

b. CLB and IOB Placement of GOSC Macro

Figure 3. Liquid Crystal Display Decoder Driver

XACT Editor and Macro Support for Quick Design Entry

The logic design is accomplished by using the XACT Design Editor running in an IBM® PC/AT™ or PC/XT™. This support software allows users to edit their designs via a mouse interface and a keyboard input. The CLB and IOB elements may be configured and connected to form complete logic subsystems through the programmable interconnect. The XACT Editor allows the user to view a section of the LCA device on the display unit of the PC, for CLB and IOB placement, and distribution of interconnect. Supported by the zoom-in and zoom-out facility, and a "world view" feature (this allows the user to see the entire LCA device layout) enables a designer to "hook" subsystems of the design together. Menus are displayed at the head of the video display, and a designer may move the cursor with a mouse to select options. These include BLOCK commands enabling CLB/IOB editing; CONFIG to enable configuration of blocks; and NET and PIN to assign pins to nets, so creating the overall interconnection. This could be analagous to interconnections on a printed-circuit card, making the electrical connection between pins of 74LSXX or other types of logic devices.

The XACT editor is equipped with a support MACRO library. This library allows widely used functions to be invoked without redesign. The relaxation oscillator, used for the backplane generator, is supplied as an existing design configuration in the MACRO. The oscillator MACRO may be called by downloading the GOSC MACRO file.

When using the XACT editor the application of already existing MACRO support files can dramatically reduce the overall design cycle time. The six-digit decoder display driver is basically repetition of the same six logic subsystems, one for each digit as shown in Figure 4. The CUTMACRO feature is useful as a support facility in avoiding repetitive design operations on individual IOBs or CLBs. Once a CLB has been designed, it can be stored as a MACRO, and given a user-defined name. The designer may call that MACRO wherever the IOB or CLB needs configuration and placement. The CUTMACRO feature also supports multiple CLBs, IOBs and interconnects of subsystems. A decode section identical to the one shown in Figure 4 can be stored as a MACRO and given a user-defined name. The design for a decoder section needs only to be performed once. The remaining five sections may be called and placed using the user-defined MACRO support facility.

Widely Used XACT Functions

As described,the XACT software is menu-driven and operations supported by the design editor are listed at the head of the screen with a schematic representation of a portion of the device shown below. A mouse interface permits the designer to scan a cursor over the schematic, or choose a functional operation by "clicking" on the selected option. Following are the available options from left to right:

NET PIN BLK CONFIG SCREEN MISC PROFILE

When selected, a heading will list its menu options. The BLK command gives nine options. The most commonly used options are EDITBLK and ENDBLK, which allow the designer to select a specific logic block and enter a logic description into it. The latter command will terminate the editing process and return the designer to a world view of the LCA schematic. Details on editing CLBs are described in more detail in the CLB Configuration Section. Other commonly used commands in the BLK menu include COPYBLK, to copy a repetitive logic design into other unconfigured blocks, DELBLK to erase unwanted designs from the network; CLRBLK, to clear the contents of a block but leave its input and output pins attached to the nets; MOVBLK, move a configured block with its associated nets to another block location; and NAMEBLK, to permit the assignment of user-defined names. Each time a function is invoked, the XACT editor prompts for a response at the bottom of the screen. For example, EDITBLK would invoke a "SELECT

BLOCK" invitation. The user then "clicks on" the block he/she wishes to configure.

The CONFIG command has six subcommands which relate to the editing of a CLB. The BASE command directs which logic arrangement is to be selected: one Boolean function of four-input variables, or two functions of three-input variables, or multiplexed input selecting either one choice of two three-input variables. EQUATE is commonly used, and allows a designer to generate a Boolean equation from keyboard input and configure it to an ouput. Other choices include ORDER, to establish blank truth tables for Karnaugh map entry; CLEAR, to delete unwanted functions; and CDATA (see data), for a text description of the block configuration.

The creation of configured blocks must have the support of interconnections. The NET and PIN commands are invoked to establish the connection of CLB outputs to CLB or IOB inputs. The NET menu may be invoked to create a net onto which input pins and one output pin is listed. The ADDNET command allows a designer to add a net to the netlist, and ADDPIN allows the user to add pins to a selected net. The NAMENET subcommand in the NET menu allows the default value of netx (where x is an XACT default-assigned integer) to be overruled by a user-assigned name. This is similar to the NAMEBLK command for BLK functions. Nets can be routed, unrouted, deleted and merged by the ROUTE, UNROUTE, DELNET and JOINNET commands, respectively. Manual editing of the net is invoked by the EDITNET command. A net is established and manually interconnected to Programmable Interconnect Points (PIPs) to create a circuit. The other NET options include FLAGNET, which assigns critical or non-critical status to a net; and HIGHLIGHT/UNHIGHLIGHT commands which "bright up" or "clear" net distribution on the "world view."

The PIN commands are: ADDPIN for adding pins to nets, and CLEARPIN for removing them. The SWAPPIN allows pins to be switched without switching functionality, where SWAPSIG exchanges the logic functionality inside the CLB without switching the pins. MOVEPIN will move a pin from one location to another, and ROUTEPIN will establish an interconnection on a net.

These and other menu functions are described in greater detail in Monolithic Memories' XACT Development System manual.

Design Methodology

Figure 4 shows a block schematic of one seven-segment decoder driver. Seven logic blocks decode the four data inputs DA, DB, DC, and DD, where DA is the least significant data input. Each block has the capability of storing the data in a register inside the logic block. The backplane oscillator drives the LCD backplane for all segments. The individual segments are driven from an exclusive-'OR" (XOR) gate, which is programmed to invert the backplane waveform for an illuminated segment, and not invert for a blanked segment. The segment decoder circuits, SA0, SB0 to SG0, provide a logic LOW output for an active segment drive. Therefore, the backplane input to the XOR gate is inverted in a second logic block and provides the correct phase control for the segment driver. The configuration was repeated six times in the LCA device to support a total of six digits.

Figure 5 shows the truth table for decoding four binary inputs into a hexadecimal segment drive. DA is the least significant binary input and DD is the most significant. The hexadecimal weighting of the binary input is also shown in the table. To illuminate a zero digit, all the segments are driven active except for segment "g," for digit "one"; segments "b" and "c" are active, and for digit "two" segments "a," "b," "d," "e" and "g" are driven. The complete decode arrangement is shown at the head of Figure 6.

The Karnaugh maps, derived from the truth table shown in Figure 5 are given in Figure 6, and provide the decode equations for all the segments. Logic "one" entries represent illuminated segments and logic zeros represent extinguished segments. A greater reduction efficiency of minterm entries was achieved using reduction techniques applied to map locations containing logic zeros. This required an inverted input to the exclusive OR (XOR), polarity control gate, as shown in Figure 4. The logic equations derived from the

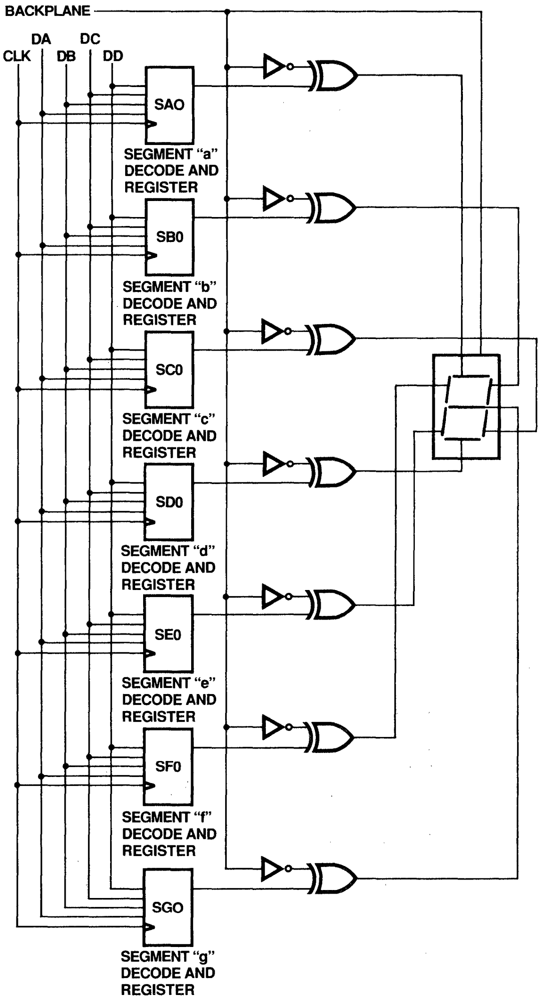
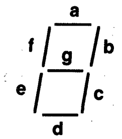

BACKPLANE
DA
DC
CLK
DB
DD
SAO
SEGMENT "a" DECODE AND REGISTER
SBO
SEGMENT "b" DECODE AND REGISTER
SCO
SEGMENT "c" DECODE AND REGISTER
SDO
SEGMENT "d" DECODE AND REGISTER
SEO
SEGMENT "e" DECODE AND REGISTER
SFO
SEGMENT "f" DECODE AND REGISTER
SGO
SEGMENT "g" DECODE AND REGISTER

Figure 4. Liquid Crystal Display Decoder Driver



BINARY ENCODED INPUT					SEGMENT LABEL						
	LSB		MSB								
HEX.	DA	DB	DC	DD	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	0	0	1	1	1	1	0	0	1
4	0	0	1	0	0	1	1	0	0	1	1
5	1	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	0	1	0	1	1	1	1	0	1	1	1
B	1	1	0	1	0	0	1	1	1	1	1
C	0	0	1	1	1	0	0	1	1	1	0
D	1	0	1	1	0	1	1	1	1	0	1
E	0	1	1	1	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

Figure 5. Liquid Crystal Display Decoder Driver

Karnaugh maps in Figure 6 could be used for Boolean design entry, the active LOW equation for segment "a" is given as:

~a = ~A*~B*C*~D + A*~B*~C*~D + A*B*~C*D + A*~B*C*D

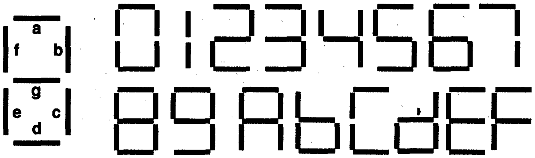
where the inputs to the CLB are A, B, C and D. The tilde sign (~) represents signal inversion, the * and + signs represent product and sum terms, respectively. The Boolean entry also supports the function XOR by the @ symbol, the function ~A*B + A*~B is condensed to A@B.

The conventional logic gate configurations are shown in Figure 7. Seven "sum of product", combinational circuits were derived from the Karnaugh maps shown in Figure 6. Logic designers familiar with PLD designs will recognize a sum of products architecture. Having generated Karnaugh maps and equations, the design can be implemented in the LCA device by using the facilities of the XACT Design Editor running in a PC/AT or PC/XT.

Binary to Seven-Segment Decoder

The display font shows the decoding function required for a binary to seven-segment hexadecimal display. A logic one represents an illuminated segment, from the table shown in Figure 5.

LCD Seven-Segment Display Driver in LCA



D,C \ B,A	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	C	D	F	E
10	8	9	B	A

For binary input ABCD the hexadecimal value of that binary weighting is entered in the correct location of the Karnaugh map.

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	1	0	1	1
10	1	1	0	1

~a = ~A*~B*C*~D + A*~B*~C*~D
+ A*B*~C*D + A*~B*C*D

(a)

D,C \ B,A	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	0	1	0	0
10	1	1	0	1

~b = ~A*B*C + A*B*D
+~A*C*D + A*~B*C*~D

(b)

D,C \ B,A	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	1	0	0
10	1	1	1	1

~c = ~A*C*D + B*C*D
+~A*B*~C*~D

(c)

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	1	1	0	1
10	1	1	1	0

~d = A*B*C + ~A*B*~C*D
+ ~A*~B*C*~D + A*~B*~C*~D

(d)

D,C \ B,A	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	1	1	1	1
10	1	0	1	1

~e = A*~B*~C + A*~D
+ ~B*C*~D

(e)

D,C \ B,A	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	1	0	1	1
10	1	1	1	1

~f = B*~C*~D + A*~C*~D
+ A*B*~D + A*~B*C*D

(f)

D,C \ B,A	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	1	1	1
10	1	0	1	1

~g = ~B*~C*~D + A*B*C*~D
+ ~A*~B*C*D

(g)

Figure 6. Liquid Crystal Display Decoder Driver

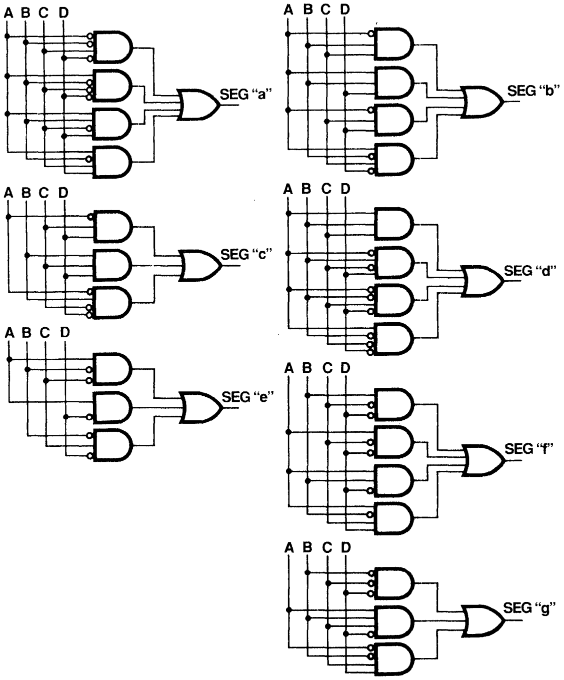


Figure 7. Liquid Crystal Display Decoder Driver

The decode logic for each segment is shown in the conventional sum of products form and was derived from the Karnaugh Maps. It is possible to encode these logic configurations in to one "CLB" for each segment. The "CLB" is capable of producing one output that is a Boolean function of up to a maximum of four inputs.

Editing the Design

The XACT Design Editor was used to create the design, and generate the configuration data, suitable for programming into an EPROM. Three programmable elements have to be considered; the IOBs, CLBs and the interconnection of IOB and CLB blocks.

CLB Configuration

Figure 8 shows the exploded view of a CLB that has been configured as the segment "a" decoder. The XACT Design Editor provides a mix of text and diagram editing supported by a keyboard and mouse interface. The logic equation is shown as text entry at the bottom of the diagram. The combinational block, AA (user defined as SA0 by invoking the NAMEBLOCK feature in XACT), has been configured as the "a" segment decoder. Each CLB can be configured as a single output function of four Boolean input variables, or two output functions of three-input variables each. Both configurations were used in this design example.

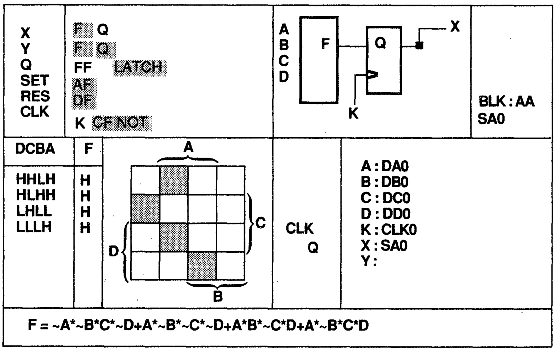
The Karnaugh map reflects the equation entered from the keyboard. An alternate way of entering configuration information may be achieved by using the mouse to select individual locations in the Karnaugh map. These locations may be "clicked" on or off, to select or deselect minterm entries. The equation relevant to the Karnaugh map entry is updated after modification. If designers have generated Karnaugh maps during the design cycle of the project, they could use this visual aid to enter and check the logic integrity of the design. Also, a truth table is available for verification purposes.

An unconfigured CLB will not display a Karnaugh map so to generate a four-input map, the instruction:

```
Config(Order(F(A(B(C(D)))))
```

is typed to create one output F that is a function of four Boolean inputs. A blank Karnaugh map will be displayed ready for editing.

The top left-hand portion of Figure 8 indicates how the CLB is configured. The Q output from the register is connected to the X output of the CLB. To activate this path, the mouse is used to select the Q option. The register could be bypassed if the F output was selected. Either register or combinational configurations may be realized. The path from the register output to the CLB X output is established by activating the PIP represented by the block tying Q to X shown in Figure 8. The alternate output, Y has the same assignment options of registered or combinational outputs. There are two configurations available to the storage element; either register or transparent latch. The register's output will change as a result of a transition being applied to its clock input. For a latch, the control input is level-sensitive requiring a logic HIGH or LOW level to distinguish between storage or a transparent mode of operation. In the CLB the clock input polarity can be selected by the NOT function in the CLK submenu. Two features that were not used in this design are the asynchronous SET and RESET functions. If required, the SET option can be invoked by selecting either the A input, or choosing a combinational output from F, which can also perform RESET. A single input to RESET the register or latch is the D input. To the bottom right of the CLB configuration diagram in Figure 8, all the inputs to, and outputs from, the CLB are listed. The net to which each CLB input or output is associated is shown after the colon. The A input is tied to the net DA0; B, C and D inputs are connected to nets DB0, DC0, and DD0, respectively. The user can assign meaningful names to nets in the design process. In this example, net SA0, is segment "a" of the least significant digit and is driven from the X output of the CLB. DA0-DD0 are the four binary inputs for the least significant digit. The clock input is assigned to the K terminal of the CLB, thus, the decoded output may be registered after a rising edge of the clock has been applied to the net CLK0.


X F Q
Y F Q
Q FF LATCH
SET AF
RES DF
CLK K CF NOT

BLK : AA
SA0

DCBA	F
HHLH	H
HLHH	H
LHLL	H
LLLH	H

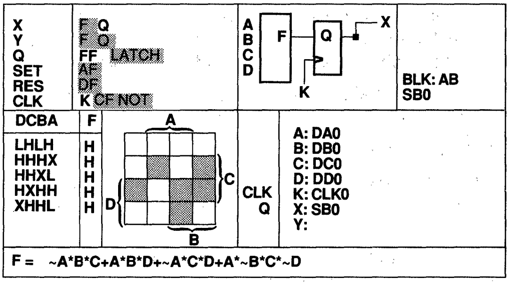
CLK
Q

A : DA0
B : DB0
C : DC0
D : DD0
K : CLK0
X : SA0
Y :

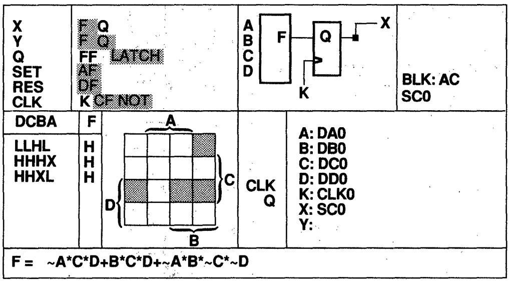
F = ~A*~B*C*~D+A*~B*~C*~D+A*B*~C*D+A*~B*C*D

CLB configuration for segment 'a' decoder.

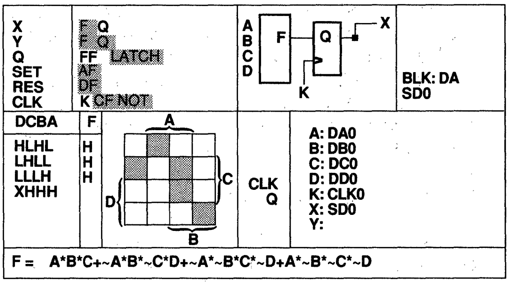
Figure 8. Liquid Crystal Display Decoder Driver



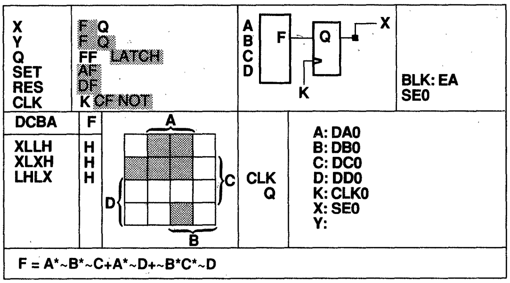
CLB configuration for segment 'b' decoder.



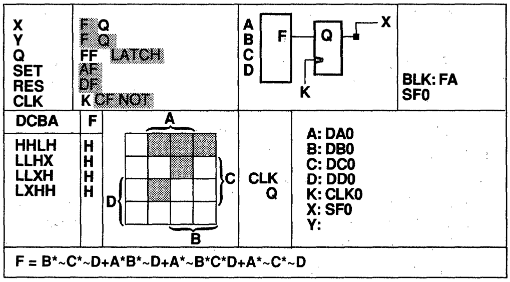
CLB configuration for segment 'c' decoder.



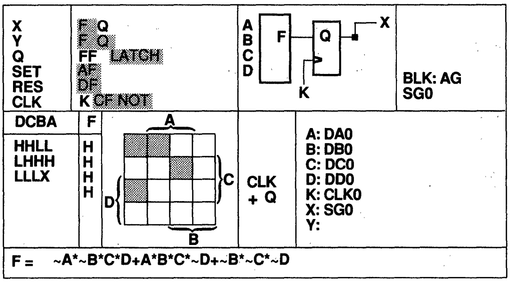
CLB configuration for segment 'd' decoder.



CLB configuration for segment 'e' decoder.



CLB configuration for segment 'f' decoder.



CLB configuration for segment 'g' decoder.

Figure 9. Liquid Crystal Display Decoder Driver

Like nets, individual blocks may be given meaningful names. The CLB situated on a grid location "AA" of the LCA device has been named SA0. Naming nets and blocks can help in debugging the final design. Figure 9 shows the configuration of the CLB elements for segments "b," "c," "d," "e," "f" and "g."

Figure 10 shows the configuration of a CLB as a combinational logic circuit which provides two outputs at X and Y. The backplane waveform is fed to the A input where it is inverted. This signal will pass through the X and Y outputs, appearing either in phase with the backplane reference or shifted by 180 degrees from it. The control inputs come from the segment decoder circuitry. In Figure 10, SA0 and SB0 are the interconnecting nets that convey control signals from the segment "a" and "b" decoders. The configuration used was two Boolean outputs as functions of three-input variables, so outputs F and G were selected.

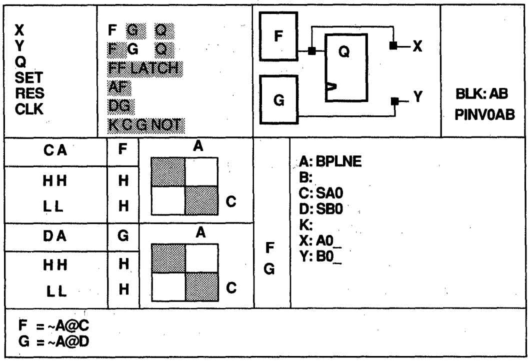
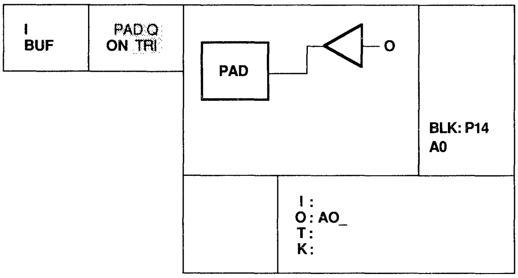


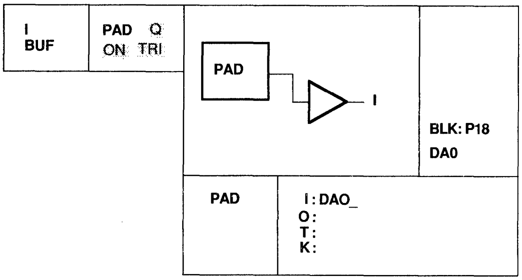
Figure 10. Liquid Crystal Display Decoder Driver

IOB Configuration

The Input/Output blocks must be configured to drive the segments, backplane, and to receive data for decoding. The segment and backplane IOBs are configured as outputs. Inputs are configured for data ports DA0-DD0, DA1-DD1, up to DA5-DD5. The IOBs are considerably less complex than the CLBs, having interface functionality rather than Boolean logic functionality. Each IOB is capable of being programmed as input, output, bidirectional, with HIGH-Z output, and registered input. Individual or composite I/O functions may be selected. Figure 11a shows an output buffer, the input of which is driven from net A0_. Net A0_ is driven from the X output of the CLB featured in Figure 10. The designer has some control over the placement of IOBs around the perimeter of the LCA device, and can use the BLKMOVE commands to optimize pad layout for the best printed circuit design. Figure 11b shows an IOB that has been configured to receive data. The buffer is an input driving net DA0_.



a. Liquid Crystal Display Decoder Driver



b. Liquid Crystal Display Decoder Driver

Figure 11. Input/Output Block Configuration

The IOB has been configured as an output buffer to drive the segment "a" of the least significant digit. The block has been assigned the name A0. the output buffer has been turned on and the input to that buffer has been configured to net AO_. All the segments and the backplane of the LCD have been assigned an output buffer.

The data path to the LCA device for binary data is via IOBs configured as input buffers. The binary inputs DA0, DB0, DC0, and DD0 for the least significant digit and the configured IOBs are assigned user-defined names DA0, DB0, DC0, and DD0. The input nets are DA0_, DB0_, DC0_, and DD0_ respectively. This is repeated for digits 1, 2, 3, 4, and 5.

XACT Supports Design Rule Checking

During the design of the LCD decoder driver circuit the Design Rule Checker (DRC), was invoked to verify that no fundamental design rules were being violated. Errors can easily occur, especially for the uninitiated user. Errors such as two CLB outputs driving one net are usually caught dynamically. Also, a net listing of inputs that are not driven by an output would constitute an error. A general DRC run will list all design violations, errors and warnings. Warnings such as an assigned CLB output that is not connected to a net. This information can be sent to a line printer and the resulting list of Errors and Warnings can be used for design correction. Invoking DRC will check blocks, interconnect and nets. Also, prior to using the MAKEBITS command (for the eventual MAKEPROM) software, the DRC is invoked to trap any possible design violations. Of course, the design rule checker does not screen the design for logic function integrity. An additional software package is available in P-SILOS™ as a logic, and timing simulator. Inputs may be activated by HIGH, LOW, HIGH-Z levels etc., while output logic levels may be listed during the simulation run.

Conclusion

The design was modified for a decimal decoder display driver. Figure 12 shows the modifications necessary to each segment decoder. The choice of a latched version of the display driver was developed as an alternative to the registered type. In each registered CLB, the option was changed for a latched option. By removing the backplane oscillator and making the net BPLNE an input which is driven HIGH or LOW, high efficiency LED displays of common anode or cathode may be driven. The designs developed are as follows:

XDES01.LCA REGISTERED HEX DECODER DRIVER.

XDES02.LCA REGISTERED DEC DECODER DRIVER.

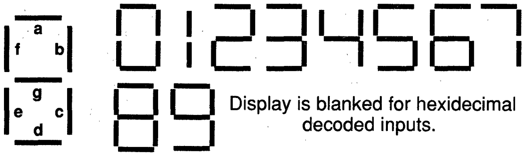
XDES03.LCA LATCHED HEX DECODER DRIVER.

XDES04.LCA LATCHED DEC DECODER DRIVER.

These designs are available as bit patterns for programming EPROMs on request.

The Print World of the design DES01.LCA shows the placement of CLBs, IOBs and the routing for the final design. This can be used as a reference for wiring the device into a circuit. Pin 1 is shown at the top center of the diagram and is the GND connection. The pins are numbered counter-clockwise from that reference pin. For example DB3 is connected to pin 2 and D2 connected to pin 3 etc.

LCD Seven-Segment Display Driver in LCA Binary Coded Decimal



Display is blanked for hexidecimal decoded inputs.

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	0	0	0
10	1	1	0	0

~a = C*D + B*D + ~A*~B*C*~D + A*~B*~C*~D

(a)

D,C \ B,A	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	0	0	0	0
10	1	1	0	0

~b = C*D + B*D + ~A*B*C* + A*~B*C

(b)

D,C \ B,A	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

~c = C*D + B*D + ~A*B*~C

(c)

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	0	0	0	0
10	1	1	0	0

~d = C*D + B*D + A*B*C + ~A*~B*C + A*~B*~C*~D

(d)

D,C \ B,A	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	0
10	1	0	0	0

~e = A + ~B*C + B*D

(e)

D,C \ B,A	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	0	0	0	0
10	1	1	0	0

~f = B*D + A*B + C*D + B*~C + A*~C*~D

(f)

D,C \ B,A	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	0	0	0
10	1	1	0	0

~g = C*D + B*D + A*B*C + ~B*~C*~D

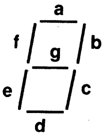
(g)

Karnaugh maps for binary coded decimal drivers. A logic zero represents an extinguished segment.

Figure 12. Liquid Crystal Display Decoder Driver—For Binary Coded Decimal Output

BCD to Seven-Segment Decoder

The table below shows the decoding function required for a BCD to seven-segment hexadecimal display. A logic one represents an illuminated segment.



BINARY ENCODED INPUT					SEGMENT LABEL						
	LSB		MSB								
HEX.	DA	DB	DC	DD	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	0	0	1	1	1	1	0	0	1
4	0	0	1	0	0	1	1	0	0	1	1
5	1	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	0	1	0	1	0	0	0	0	0	0	0
B	1	1	0	1	0	0	0	0	0	0	0
C	0	0	1	1	0	0	0	0	0	0	0
D	1	0	1	1	0	0	0	0	0	0	0
E	0	1	1	1	0	0	0	0	0	0	0
F	1	1	1	1	0	0	0	0	0	0	0

Figure 12. Liquid Crystal Display Decoder Driver

Print World: DES01.LCA (2018PC84-50), XACT 1.30, 15:12:14 JUL 1, 1987   Print World: DES01.LCA (2018PC84-50), XACT 1.30,

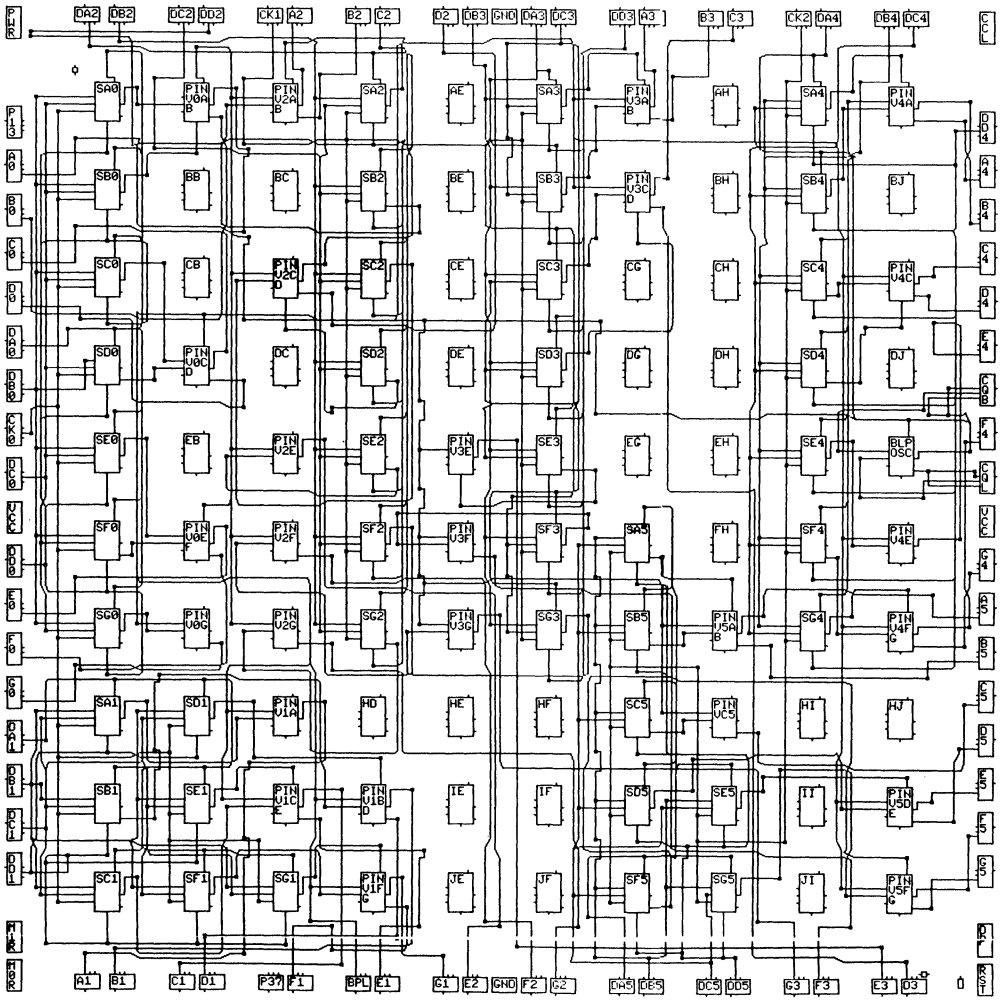


Figure 12. Liquid Crystal Display Decoder Driver

LCA Counter Applications

Chris Jay and Karen Spesard

Abstract

Counters are used in many logic systems to control and synchronize events. Essentially, counters have one thing in common, they are all state machines. State machines, unlike basic combinational functions, require registers with feedback. The design creates an output that is a function of the previous state of the registers in the machine, and in some cases a function of other combinational inputs as well. Listed below are some of the systems that would use various types of counters.

- Direct Memory Access (DMA) Controllers
- Video RAM Refresh
- Dynamic RAM Control Refresh Counters
- Event timing
- Sequence Controllers
- Disc Controllers
- Status Sequencers

The type of counter chosen will depend upon the application. The simplest type of counter would be a free-running type as used for refreshing DRAM or Video RAM. Counter types vary, depending on application from the simple free-running type to loadable binary up/down counters which would be used in applications such as DMA control.

To design efficient counters in the LCA device, designers must consider the desired performance while keeping in mind the available logic and interconnection resources.

Familiarization with the different counter types and their characteristics will enable designers to choose the best counter for a specific application.

LCA Counter Applications

Chris Jay and Karen Spesard

Introduction

The LCA device has an architecture that is very well suited to the development of some counters. Designers already familiar with state machine applications in Programmable Logic Devices (PLDs) will probably be familiar with how to design counters in Programmable Array Logic (PAL®) and/or Programmable Logic Arrays (PLA). The differences between the structure of the PLA or PAL device and the Monolithic Memories' Logic Cell™ Array (LCA) impose both restrictions and freedoms in certain types of state machine design.

Ample width of the PLD "AND" gate means that counter depth can be limited only by the number of registers in the device. This applies especially in the PAL24X family that is specifically designed for applications in high-performance binary counting. In the LCA device, the maximum number of Boolean inputs to any CLB is four. This limiting factor is compensated by the high number of CLBs present in the device architecture. The design philosophy used in an LCA device for developing state machines is different from that of a PLD, but not less effective. Parallel counter architecture, and lookahead-carry techniques can be employed to create machines capable of medium performance.

Counters Using Shift Registers

The type of state machine well suited to an LCA device is based on a shift register. These types may be subdivided into different categories. In each state machine, there are "n" general registers, and the number of states that can be reached vary with counter type. The number of useable states are listed below with the counter type:

1) Johnson counters, 2n, states.

2) Linear feedback shift registers, with 2^n - 1 states.

3) Modified linear feedback shift register, with 2^n states.

These state machines are based on shift registers using the absolute minimum amount of feedback, and this is applied only to the least significant register in the chain. The CLBs may be located at close proximity in the LCA device and benefit from the smallest propagation delays provided by local interconnection. The disadvantage of the shift register structure is the inability to count through the conventional binary sequence. This might or might not be of any concern, depending on the system application.

The Johnson Counter

All of the states of a six-bit Johnson counter are listed in Table 1. The "D"-type register chain shown in Figure 1 illustrates the schematic diagram of the circuit. The output from Q0 feeds directly to D1, and Q1 to D2, and so on. If these registers are configured in adjacent CLBs, direct interconnect can be used to link each one. The output of the final register Q5 is inverted and fed back to the D0 input of the first register. If the entire structure is configured in a column or row in the LCA device, a long line should be used to convey the feedback signal back to the D0 input of the least significant register. Minimum propagation delay is achieved by using long line interconnect. The advantage gained with the small propagation delay of direct short interconnections between adjacent CLBs is not lost by the use of this long feedback path.

The counter design shown in Figure 1 uses six registers and the maximum number of states reached is 12, two times the number of registers. A binary counter with six registers could reach sixty-four individual states but would need combinational feedback to each register. Feedback propagation delay could degrade the potential clocking speed of the counter. The Johnson counter, with its decreased number of states, should be used whenever maximum operational performance is needed.

STATE	Q0 Q1 Q2 Q3	Q4 Q5	HEX
0	0 0 0 0	0 0	0
1	1 0 0 0	0 0	1
2	1 1 0 0	0 0	3
3	1 1 1 0	0 0	7
4	1 1 1 1	0 0	F
5	1 1 1 1	1 0	1F
6	1 1 1 1	1 1	3F
7	0 1 1 1	1 1	3E
8	0 0 1 1	1 1	3C
9	0 0 0 1	1 1	38
10	0 0 0 0	1 1	30
11	0 0 0 0	0 1	20

Table 1. Truth Table of the Johnson Counter Counter Applications

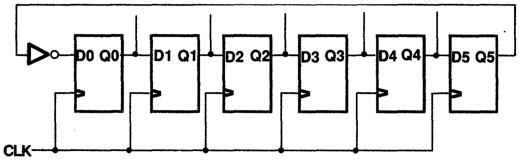


Figure 1. Johnson Counter

PAL® is a registered trademark of Advanced Micro Devices.
Logic Cell™ and XACT™ are trademarks of Xilinx, Inc.
P-Silos™ is a trademark of SimuCad Corporation.

IBM® is a registered trademark of International Business Machines Corporation.
PC™, PC-AT™, and PC-XT™ are trademarks of International Business Machines Corporation.

Linear Feedback Shift Register

Figure 2 shows a modification to the Johnson counter. In this three-bit counter, six states are reached. The eXclusive NOR feedback from register outputs Q1 and Q2 gives a HIGH input to D0 when both are HIGH or LOW. Again the design is based on a shift register, and this time the feedback is applied to the least significant register in the chain. The Linear Feedback Shift Register or LFSR shown in Figure 2 is implemented in three CLBs for small state machine designs. CLB placement and routing could be optimized for speed and performance. The LFSR counter would have a "stuck" state if all of the registers were set to a logic HIGH. The feedback input to D0 would remain HIGH and the XNOR inputs from Q1 and Q2 would remain stuck at a logic HIGH. The device has 2^n - 1 states because in normal counting the "stuck" state cannot be entered. Table 2a shows the sequence and truth table of a three-register, free-running LFSR.

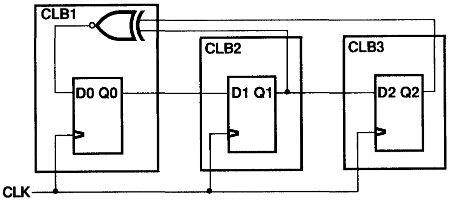


Figure 2. Linear Feedback Shift Register (LFSR)

Modified Linear Feedback Shift Register

The "stuck state" is included in the truth table shown in Figure 2b. If after the count of 3, the state machine could be modified with gating to accommodate the "stuck state" of 7, then an additional state could be used. Shifting a HIGH rather than a LOW into the least significant register would generate state 7, and all of the possible states could be reached. Table 2b shows the new entry in the truth table, and the Karnaugh map in Figure 3b shows the state assignment entry. For example, the map location Q0 = 0, Q1 = Q2 = 1 is the binary code 6. The State Diagram of the counter (Figure 3a), which is derived from the truth table (Table 2b), shows the sequential flow of each state into the next. From information in the State Diagram and State Assignment Map, a State Excitation Map may be developed as shown in Figure 3c. The next state, or excitation state from STATE 6 in Figure 3a is STATE 5. The corresponding entry of 6 in the Assignment Map is replaced by 5 in the Excitation Map and represents the transition from 6 to 5. Figure 3c is developed into Figure 3d by converting the entries to binary notation and placing the condition of the least significant bit into the Karnaugh map shown in Figure 3d. Minimization gives a Boolean equation:

Q0 := ~Q1*~Q2 + Q0*~Q2 + ~Q0*Q1*Q2 (1)

Equation (1) is the functional input to the least significant register in the chain. The other registers in the chain require only the direct inputs from the preceding registers, so no additional minimization is required. Figure 4 shows the final gate implementation which can be incorporated into CLB 1 replacing the XNOR gate.

The advantages of using every possible state in the counter avoids the possibility of a stuck state which, if entered, will prevent the entire design from functioning.

A very important consideration of the LFSR in CLB-intensive designs is the ability to use IOB registers in the counter. While the IOBs have no logic functionality they do have registers. The LFSR is based on the shift register, so if CLB resources run out, IOBs could be used.

COUNT	Q0 Q1 Q2
0	0 0 0
1	1 0 0
3	1 1 0
6	0 1 1
5	1 0 1
2	0 1 0
4	0 0 1

Table 2a. Truth Table of a Linear Feedback Shift Register

COUNT	Q0 Q1 Q2
0	0 0 0
1	1 0 0
3	1 1 0
7	1 1 1
6	0 1 1
5	1 0 1
2	0 1 0
4	0 0 1

Table 2b. Truth Table of a Modified Linear Feedback Shift Register

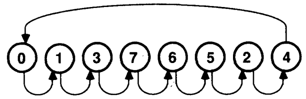


Figure 3a. State Diagram for Modified Linear Feedback Shift Register

Q1

	0	2	6	4
Q0	1	3	7	5

Q2

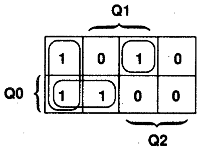
Figure 3b. State Assignment Map

Q1

	1	4	5	0
Q0	3	7	6	2

Q2

Figure 3c. State Excitation Map



```
Q0 := ~Q1*~Q2 + Q0*~Q2 + ~Q0*Q1*Q2
```

Figure 3d.

From the state excitation map, the condition of the least significant register Q0 is entered. Minimization gives the equation required for Q0 IN as a Boolean function of Q0, Q1 and Q2. When implemented, the modified LFSR will count through all states.

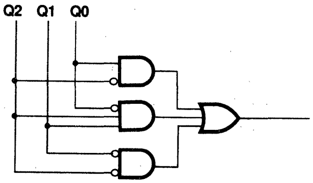
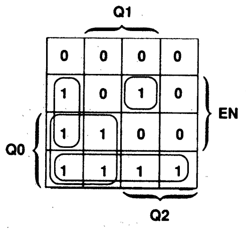


Figure 4.

If this circuit is used to replace the exclusive NOR gate in CLB1 of Figure 2 then the count shown in Table 2b will be realized enabling a count through 2^n states.

Provisions for Deeper Counting

Deeper counters may be made from cascading 3-bit LFSRs. The same three-bit module may be repeated any number of times but the original circuit needs modification. When cascading counters it is essential to use an ENABLE or CARRY input to the next stage. The Karnaugh maps, shown in Figures 3b and 3c, represent State Assignment and State Excitation, respectively. The Assignment Map can be considered as identifying HOLD conditions while the Excitation Map indicates count activity. The Karnaugh map in Figure 5 merges the information contained in both Figures 3b and 3c, and an additional ENable input qualifying COUNT and HOLD activities. The registers will count if EN is HIGH and HOLD when it is LOW. The Karnaugh map in Figure 5 was developed to derive the Boolean equation for the least significant register in the chain. The other registered cells need modifying to incorporate an enable input. Table 3 shows a truth table of the current register contents Qn with the EN input B and the input from the next least significant register A. The required output Q_{n+1} is given as a HOLD condition if B = LOW, and when HIGH, data from the A input will be clocked into the register. For each of the three registers, an EN input will enable count activity to allow cascading of 3-bit counter modules.

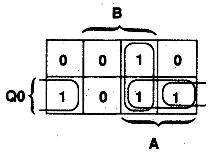


```
Q0 := Q0*~EN          ;HOLD FUNCTION.
   + Q0*~Q2*EN        ;COUNT
   + ~Q1*~Q2*EN       ;COUNT
   + Q2*Q1*~Q0*EN     ;COUNT.
```

Figure 5. Incorporating a Count Enable Input

Q_n	A	B	Q_{n+1}
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

Table 3.



General shift register with count enable. A = input from the next least significant register and B = enable count. Q is the state of the internal register. This Karnaugh Map has been developed from Table 3.

Q := Q*~B + Q*A + A*B

Figure 6.

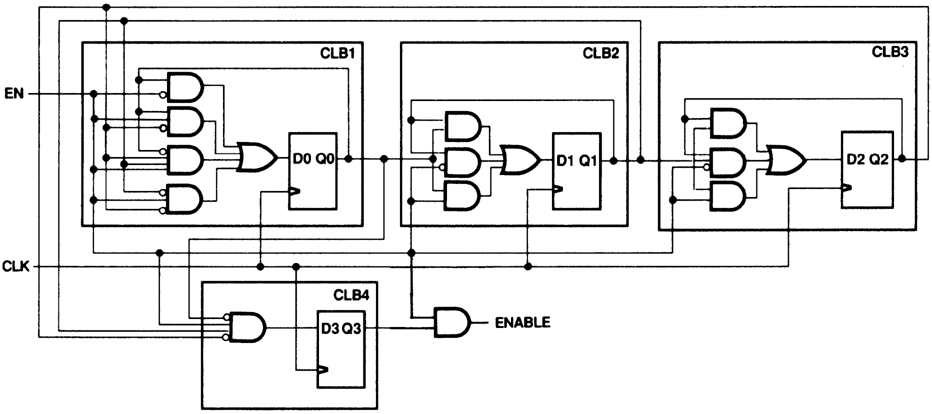
Propagation Carry

To link counter modules together, it is necessary to pass a count enable signal from one module output to the next input. Figure 7 shows the configuration of CLBs 1 to 3 linked to CLB 4, which is configured as a carry generator providing an enable output to the next modified LFSR. The next counter module will not increment until it receives an assertion on its count enable input. Figure 8 shows the Karnaugh map from which the lookahead-carry output was generated. The penultimate count in the sequence, as shown in Figure 2a is two. This count is decoded in the CLB 4 where the registered output is passed to the next counter module and delayed by one clock cycle. The concept of decoding the penultimate count, and delaying it by one clock cycle provides a fast method for propagating the lookahead carry signal. If the ultimate count is decoded by a combinational circuit, the next counter module will have to wait for the carry to propagate through logic gates before the next clock pulse can be applied. Registering the next-to-last count allows premature carry propagation, providing the carry enable to be set up prior to the next clock edge.

Figure 9 shows two identical LFSRs linked by a carry generator's CLBs 5,6 and 7 and a propagate carry configuration in CLB 8. The propagate carry circuit is a combinational circuit that enables the generated carry output, from CLB 4 to the ENB output, as shown in Figure 9. A third set of modified LFSRs may be added if a synchronous nine-bit counter is required.

Applications for a nine-bit counter might include a refresh counter for 256 K Dynamic RAMs. The EN input to CLB 1 can be used to hold off count increment activity. If an LFSR with $2^n - 1$ states were used, then one row in the DRAMs would never get refreshed, so the modified LFSR would be the appropriate choice. Moreover, the count sequence is not important because it does not have to be a binary code, just as long as the DRAMs are refreshed at regular intervals for every row.

Another application might be as a counter in a video controller. The system could count through the required states, and additional gating could be used to generate line and frame sync pulses at certain states during the count sequence.



ENABLE = EN*~Q0*Q1*~Q2

Figure 7.

		Q0			
	0	0	0	0	
	0	0	0	1	EN
Q2	0	0	0	0	
	0	0	0	0	
			Q1		

Figure 8. To Enable a Second Stage as a Synchronous Counter a Carry Output Must Be Generated from the First Three Registers. The Enable Output from CLB 4 Is Decoded from the Penultimate Count and Clocked Through the Register. When the Ultimate Count Is Reached the Enable Input to the Next Stage Is Asserted.

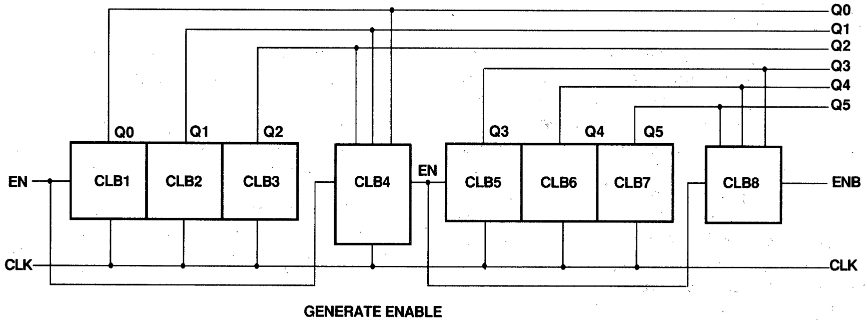


Figure 9. Two LFSRs Are Linked by the Propagate Enable Logic Contained in CLB4. CLB5, 6 and 7 Contain the Same Logic as CLB1, 2 and 3 to Create a Modulo 6 Counter with Sixty-four States. Propagate Enable is Decoded from the Ultimate Count of the Second LFSR Stage. It Can Be Used as the Enable Input to a Third LFSR Stage to Make a 9-Bit Counter.

Design of a Four-Bit LFSR

If a four-bit free-running counter was required for standalone count applications, then a design can be realized using four CLBs. The truth table for a modulo four modified LFSR is shown in Table 4. The state assignment map in Figure 10 is developed into the state excitation map shown in Figure 11. The least significant bit of the hexadecimal data is entered into the Karnaugh map as shown in Figure 12. Minimization is performed to develop the equation for the least significant register input. The actual circuit implementation is complete in four CLBs as shown in Figure 13.

	REGISTER				
STATE	Q 0	Q 1	Q 2	Q 3	CODE
0	0	0	0	0	0
1	1	0	0	0	1
2	1	1	0	0	3
3	1	1	1	0	7
4	1	1	1	1	F
5	0	1	1	1	E
6	1	0	1	1	D
7	0	1	0	1	A
8	1	0	1	0	5
9	1	1	0	1	B
10	0	1	1	0	6
11	0	0	1	1	C
12	1	0	0	1	9
13	0	1	0	0	2
14	0	0	1	0	4
15	0	0	0	1	8

Table 4.

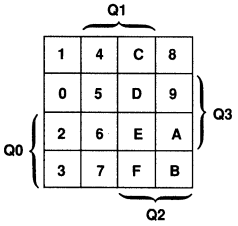


Figure 10. State Assignment Map

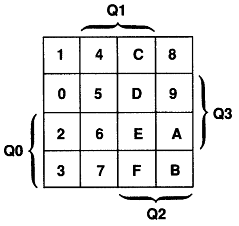


Figure 11. State Excitation Map

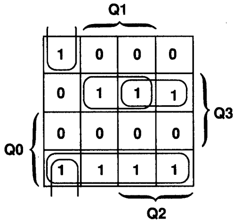


Figure 12. State Excitation Map for the Least Significant Register

Q0 := Q0*~Q3 +~Q0*Q1*Q3 +~Q0*Q2*Q3 +~Q1*~Q2*~Q3

The Binary Counter

The number of states available in a binary counter is 2^n, making it one of the most register-efficient types of counters. The three main categories of binary or weighted counters are ripple, ripple-carry, and lookahead-carry. As the amount of "look-ahead" logic decoded at each counter stage increases, the performance of the binary counter also increases. As a result, lookahead-carry counters are the fastest of the three categories of binary counters. On the other hand, the more "look-ahead" logic there is, the more decoding is needed. Increased decoding requires extra logic and routing resources which may be disadvantageous if these resources become scarce.

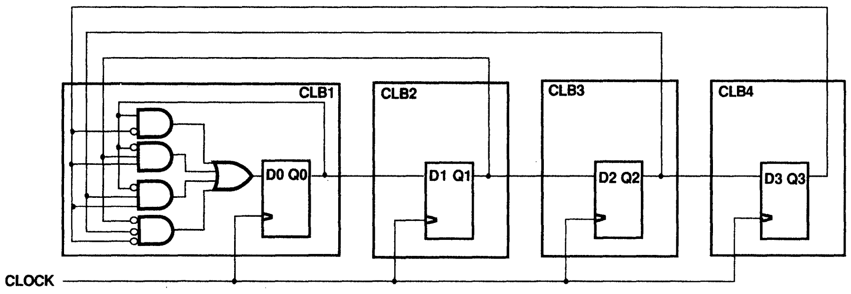


Figure 13. Modified LFSR Modulo 4 Counter

The Ripple Counter

Ripple counters are asynchronous in nature and do not generate carry signals. When the output of each counter stage clocks the next stage on the negative clock transition, a ripple effect is induced (thus the name). A schematic representation of a six-bit ripple counter is shown in Figure 14. One of the benefits of a ripple counter is that it requires few resources. Only one CLB per counter bit is needed to implement the counter and routing is simple regardless of the counter length. The tradeoff for this simplicity, however, is that ripple counters cannot be modified to be loadable or up/down which restricts their operation to be free-running, and they have lower performance.

The overall performance of a ripple counter degrades with each counter bit by one CLB delay time. This can be shown by the equation below:

Ripple Counter = N * (Clock to Output Delay) (2)
Clock Period

where N = the number of ripple counter flip-flops

The overall counter clock period must therefore be greater than or equal to the total cumulative CLB delay.

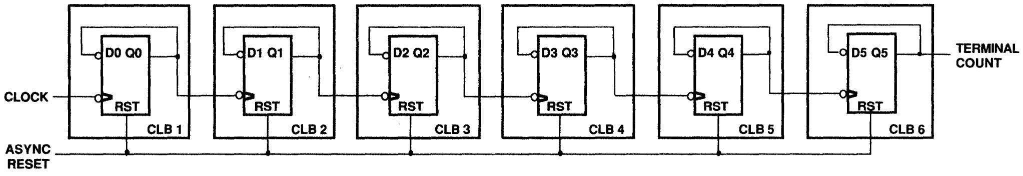


Figure 14. Schematic of a 6-Bit Binary Ripple Counter. Ripple Counters Are Easy to Design and Are Cascadable to Nearly Any Length. They Are, However, Asynchronous and Are Not Recommended for Most Designs.

The Ripple-Carry Counter

The ripple-carry counter is similar to the ripple counter. The ripple-carry counter has carry signals, however, whereas the standard ripple counter does not. Each carry signal propagates to the next counter stage to produce the counting sequence. This cascaded connection is called ripple-carry.

The ripple-carry counter differs from the ripple counter in the respect that it is synchronous in operation. Because of this, the ripple-carry counter provides more reliable operation and better performance. Performance still degrades with each additional counter stage though, due to the inclusion of combinational logic from the previous counter stage. Like the ripple counter, the ripple-carry counter requires only one CLB per counter bit and is easily cascadeable.

One example of a four-bit binary ripple-carry counter is shown in Figure 15. Here, a modulo-16 counter with count enable (CE) and reset is built with four CLBs and three levels of ripple logic. The same counter could be designed with two levels of ripple logic, if the two-input AND gate in CLB2 becomes a three-input AND gate, and the signals CE and Q0 from CLB1 were carried over to CLB2 and ANDed with Q1.

Another four-bit counter, this time with parallel enable (load), count enable, and reset, is shown in Figure 16. Here, just two ripple levels of logic were designed in, using two C8BCP MACROS from Monolithic Memories' Macrocell Library. Six CLBs, two more than the previous four-bit counter, were used in this design because of the addition of parallel enable circuitry and the reduction of ripple-carry logic levels. The Karnaugh map corresponding to the registered CLBs is shown in Figure 17.

Sometimes a designer will want to carry-over a counter implementation he or she was previously using to the LCA device. Consider an eight-bit ripple-carry counter with reset, built with two typical 74-series TTL devices: two 74-161's shown in Figure 18. Notice that the data bus as well as the LD and CET pins are not being utilized. This is wasteful. Obviously, the two 74-161 macros, available from Monolithic Memories' LCA Device Macrocell Library, requiring sixteen CLBs would also be inefficient. An equivalent counter, shown in Figure 19, requires only ten CLBs. It is built with two C8BC-rd MACROS and one C4BC-rd MACRO and contains three ripple-carry delay levels. It can be seen that implementing just the counter functions desired, instead of including additional functions not required in the counter, allows designers more control over their design. The designer can then better optimize the design for speed and performance while minimizing CLB and routing resources.

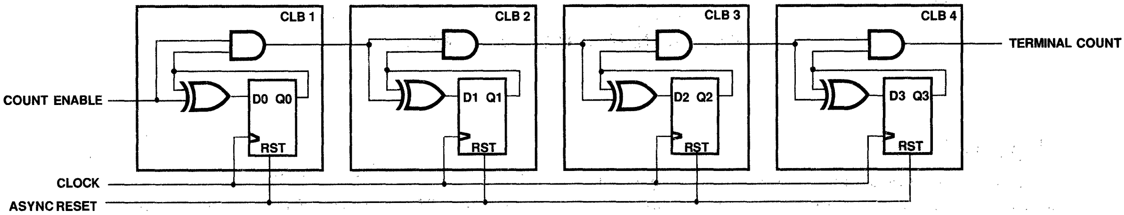


Figure 15. Schematic of a 4-Bit Ripple-Carry Counter. Although the Ripple-Carry Counter Is Synchronous, the Carry Input to the Next Stage Is Conveyed Through Combinational Logic. The Propagation Delay Due to this Logic Must be Taken into Account.

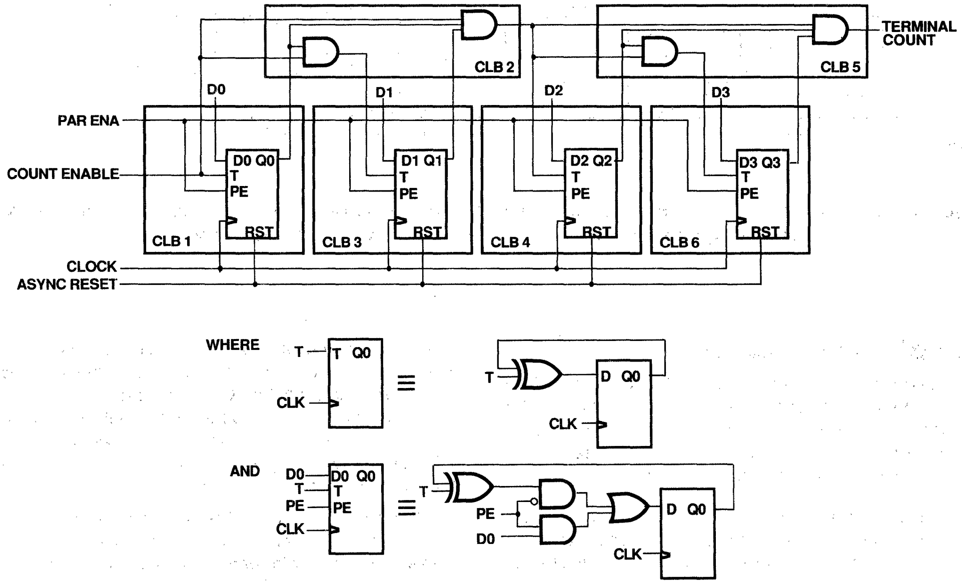
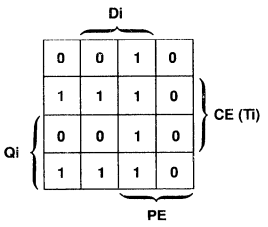


Figure 16. Schematic of a 4-Bit Ripple-Carry Counter. With Parallel Enable, Count Enable, and Reset, this Counter Only Contains Two Levels of Ripple-Carry Logic.



0	0	1	0
1	1	1	0
0	0	1	0
1	1	1	0

Figure 17. Karnaugh Map for the Registered CLBS in Figure 16.

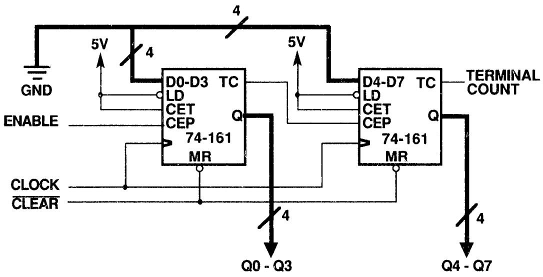


Figure 18. Schematic of an 8-Bit Counter with Reset Using Two 74-161 TTL Devices. Since All of the Available Logic Is Not Utilized, the Counter Should Be Designed More Efficiently for Use in the LCA Device.

The Lookahead-Carry Counter

When the performance of binary ripple and ripple-carry counters is not adequate, synchronous binary lookahead-carry counters can be a solution. A lookahead-carry counter incorporates all the previous counter outputs into a single lookahead-carry signal for each counter stage. This logic reduction minimizes the overall delays of the design which makes the performance of lookahead-carry counters the highest among binary counters.

With lookahead-carry counters, the complexity of the design increases with each counter bit as the decoding inputs become ever wider. As a result, these counters usually need more CLBs and routing resources to implement than other binary counters. The equations which characterize an n-bit lookahead-carry counter with count enable and parallel enable are listed below:

Q0 := ((/PARENA * CE) @ Q0) + (PARENA * DO)

Q1 := ((/PARENA * CE * Q0) @ Q1) + (PARENA * D1)

Q2 := ((/PARENA * CE * Q1) @ Q2) + (PARENA * D2) . . .

and

Qn := ((/PARENA * CE * Q1 * ... * Qn-1) @ Qn)
+ (PARENA * Dn)

To decrease the complexity of the counter, one or more of the control signals can be removed.

For a ten-bit lookahead-carry counter with reset, a minimum of fourteen CLBs are needed as shown in Figure 20. The logic for the counter was partitioned as follows:

CLB 1	Q0 := CE @ Q0
CLB 2	Q1 := (CE * Q0) @ Q1
CLB 3	Q2 := (CE * Q0 * Q1) @ Q2
CLB 4	Q02 = Q0 * Q1 * Q2 * CE
CLB 5	Q3 := Q02 @ Q3
CLB 6	Q4 := (Q02 * Q3) @ Q4
CLB 7	Q5 := (Q02 * Q3 * Q4) @ Q5
CLB 8	Q35 = Q3 * Q4 * Q5
CLB 9	Q6 := (Q02 * Q35) @ Q6
CLB 10	Q36 = Q3 * Q4 * Q5 * Q6
CLB 11	Q7 := (Q02 * Q36) @ Q7
CLB 12	Q8 := (Q02 * Q36 * Q7) @ Q8
CLB 13	Q38 = (Q36 * Q7 * Q8)
CLB 14	Q9 := (Q02 * Q38) @ Q9

The first seven bits of the counter were decoded in a similar fashion to the counter in Figure 19. The next three bits continue to minimize the amount of ripple-carry logic implemented to make it a lookahead-carry counter.

When routing a design such as this in the LCA device, it is best to place the CLBs lengthwise. Then, if the high fan-out output signals for Q02 and Q36 CLBs can be routed through "long line" interconnects, the routing-dependent delays can be reduced allowing the counter to perform at its maximum potential speed.

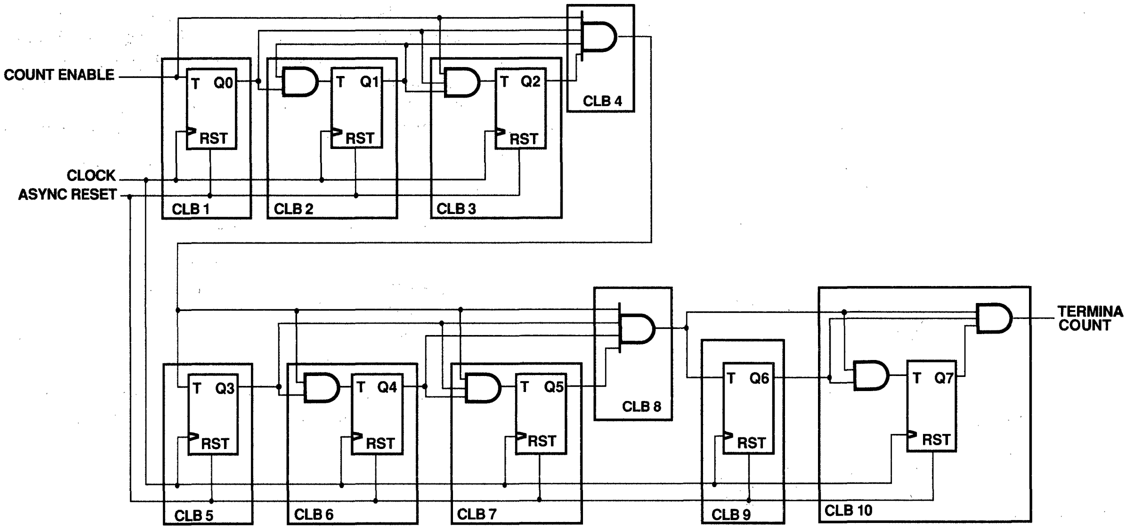


Figure 19. An Efficient Implementation of an 8-Bit Counter with Reset. This Alternative Performs Only the Desired Function While Maximizing CLB and Routing Utilization.

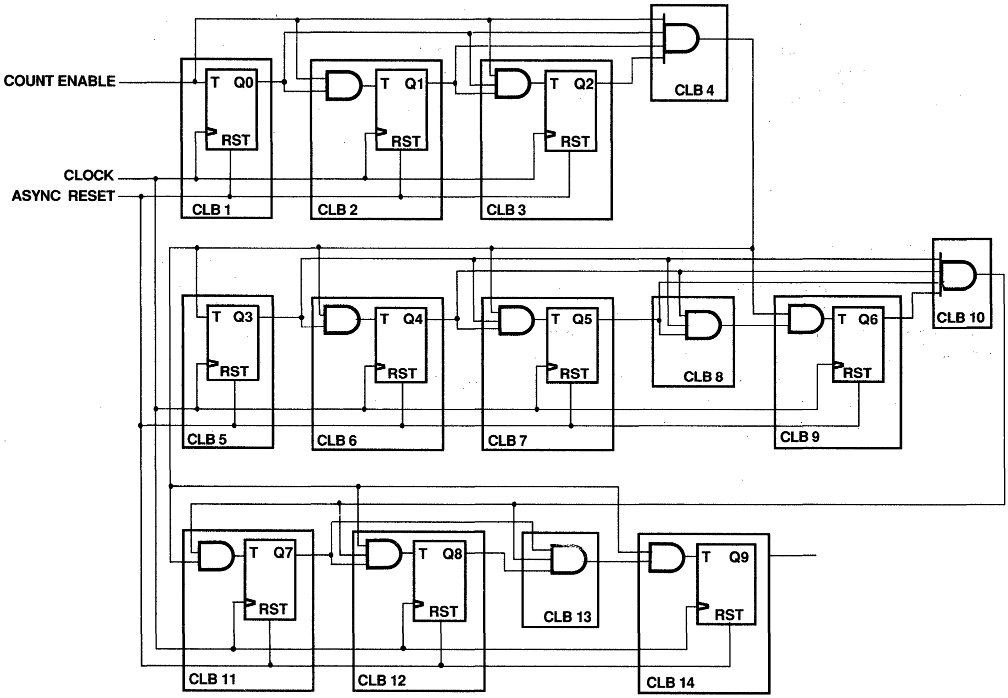


Figure 20. Schematic of a Synchronous Counter with Reset as in Figure 19, but with Lookahead-Carry Logic. As More Counter Bits Are Added, the Design Becomes Increasingly Complex Due to Wider Gating.

The Up/Down Counter

The up/down counter can be a very useful device. For instance, status counters or address counters employed in DMA systems, dual slope integrators, and delta modulation systems usually use up/down counters. In operation, an up/down counter will count UP when all previous counter bits are HIGH and will count DOWN when all previous counter bits are LOW. Thus, in an up/down counter, each register output in a CLB will toggle when:

- Q0 through Qn-1 are HIGH and the counter direction is UP, or
- Q0 through Qn-1 are LOW and the counter direction is DOWN.

This translates to:

Qn := Qn @ ((Qn-1 * Qn-2 * ... * Q1 * Q0 * UP) +
(/Qn-1 * /Qn-2 * ... * /Q1 * /Q0 * /UP)). (3)

Design of the counter can become more complicated by adding new features to it such as count enable, reset, or load (parallel enable). Because of this, it is best to limit the number of control signals which would require more complex logic. For example, a counter which only needs to be reset during initialization will not need additional reset capability since all registers are reset upon initialization of the LCA device.

Simple ripple-carry binary up/down counters require 2 * N - 4 CLBs for implementation where N is the number of counter bits. For example, eight CLBs would be needed for a six-bit up/down counter and twenty-eight CLBs would be needed for a 16-bit up/down counter. An example of a six-bit ripple-carry up/down counter configured in CLBs is given in Figure 21.

The general equations used for this ripple-carry counter design were derived from equation 3 and are as follows:

Running Total of CLBs	Equations

1	Q0 := /Q0
	C2X = UP * Q0 * Q1

2	Q1 := Q1 @ ((UP * Q0) + (/UP * /Q0))
	C2Y = /UP * /Q0 * /Q1

3	Q2 := Q2 @ (C2X + C2Y)
4	Q3 := Q3 @ ((Q2 * C2X) + (/Q2 * C2Y))

5	C3X = C2X * Q2
	C3Y = C2Y * /Q2

6	Q4 := Q4 @ ((Q3 * C3X) + (/Q3 * C3Y))

7	C4X = C3X * Q3
	C4Y = C3Y * /Q3

8	Q5 := Q5 @ ((Q4 * C4X) + (/Q4 * C4Y))

To build a simple ripple-carry 16-bit up/down counter, this algorithm would be continued:

Qn := Qn @ ((Qn-1 * Cn-1X) + (/Qn-1 * Cn-1Y))

where:

Cn-1X = Cn-2X * Qn-2
Cn-1Y = Cn-2Y * /Qn-2
Cn-1X, Cn-1Y are > or = C2X, C2Y

and

C2X = UP * Q0 * Q1
C2Y = /UP * /Q0 * /Q1.

The test file that was used to verify the logic of the six-bit up/down counter in Figure 21 is shown in Figure 22. It was generated by running the SIMGEN program included with the XACT Development System. This file, with a .DAT extension, was modified to include the desired input test vectors which would be executed. Once in the P-Silos™ simulator, "IN <filename>.DAT" is entered. This command automatically inputs the netlist to the simulator. Finally, a time period is entered which informs the simulator of the length of time the simulation should run. For instance, "SIM 0 10000" could be entered. The outputs will be plotted in a table format and will list the signals specified.

An alternative method for designing up/down counters is to use lookahead-carry logic, based on equation 3. This performance-driven method, however, is quite CLB intensive due to the wide gating of input signals required for implementation. Thus, for a six-bit lookahead-carry up/down counter, sixteen CLBs would be needed, whereas for the six-bit ripple-carry up/down counter, ten CLBs were needed. Therefore, it is advantageous to use the lookahead-carry method when maximum performance is needed.

A design of an n-bit ripple-carry up/down counter that can synchronously RESET and LOAD new values into the counter also becomes more difficult. From the general expression below, derived from state tables and Karnaugh maps, any up/down counter can be designed.

Qn = (/RESET * LOAD * Dn)
@ ((/RESET * /LOAD * UP * Qn-1 * ... * Q1 * Q0)
+ (/RESET * /LOAD * /UP * /Qn-1 * ... * /Q1 * /Q0))

where RESET and LOAD are active HIGH.

There are many ways to design up/down counters. The lookahead carry method, though it consumes a great deal of resources, usually enhances performance. The best way to design the up/down counter for systems that do not require high speed, is to use the ripple-carry method. This method is usually the most straightforward and requires the least number of resources.

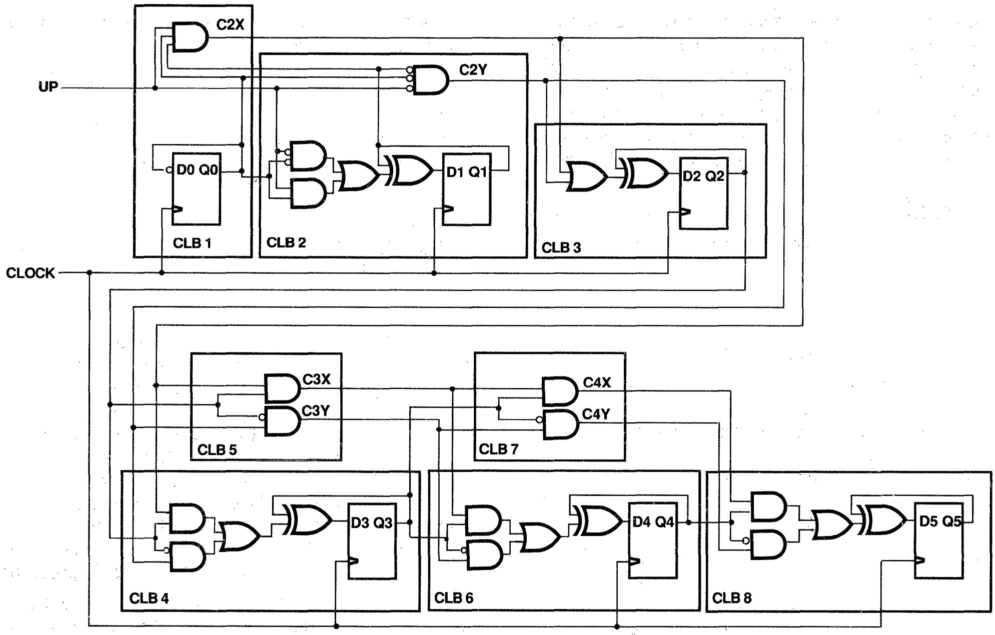


Figure 21. Schematic of a 6-Bit Up/Down Counter. Only Eight CLBs are Required for Implementation.

```
$
$ Simulation file for design 'FIG21.LCA' type '2064NL68-70'
$ Created by XACT Ver. 1.30 at 14:14:31 JUL 21, 1987
$
!INPUT FIG21.sim

$ INPUTS:
GLOBALRESET- .CLK 0 S0 1 S1 $ Initial pulse to reset latches
CLK          .CLK 0 S0 1000 S1 2000 S0 .REP 0
UP           .CLK 0 S0 128000 S1 256000 S0 270000 S1 280000 S0

 .MONITOR CLK ; UP ; Q0 Q1 Q2 Q3 Q4 Q5
 .TABLE   CLK ; UP ; Q0 Q1 Q2 Q3 Q4 Q5
```

Figure 22. Input File to P-Silos

Summary

Many ways exist for implementing counters in the LCA device. The ideal counter design for a specific application depends on the desired performance and the available logic and interconnection resources. See Table 5. For example, three main types of counters, Johnson, Linear Feedback Shift Register, and binary counters were discussed. Each of these counters utilizes different features of the LCA device but each can be optimized to perform a required function. Moreover, it was seen that by implementing only the logic necessary for a required function, resource efficiency was increased and by implementing more lookahead-carry logic or using LFSRs, performance was increased.

COUNTING METHOD	COUNTER TYPE	MODULO	CLB EFFICIENCY	ROUTING EFFICIENCY	PERFORMANCE	ADVANTAGES/ DISADVANTAGES
Non-binary	**Johnson**	2n	Poor	Excellent	Excellent with low modulo	High performance, easy routing Glitch-free decoding Low-register efficiency
	Linear feedback shift register or modified LFSR	2^n - 1 or 2^n	Good	Very good	Very good decreases with increasing modulo	Good performance at high modulos
Binary	**Ripple**	2^n	Excellent	Excellent	Poor	Low resources, easy routing but slow and asynchronous
	Ripple-carry	2^n	Very good	Good	Good	Good general binary counters
	Lookahead carry	2^n	Good but decreases with increasing modulo	Good but decreases with increasing modulo	Good for high-performance binary counters	Highest performance binary counter requires more CLB and routing resources

Table 5. Summary of Counter Types and Their Characteristics

Time Division Multiplexing with the LCA Device

C.B. Lee and Theresa Shafer

Abstract

High-speed data communication lines are efficiently utilized when low-speed signals are time division multiplexed onto high-speed lines. Time division multiplexing requires data buffering, rate adaption and data selection. The data selection fits into a single CMOS Logic Cell™ Array (LCA device). The LCA device is designed and optimized using the FutureNet® schematic capture package, Monolithic Memories' automatic place and route software, and the XACT™ Design Editor System.

Time Division Multiplexing with the LCA Device

AN-161

Introduction

The Logic Cell Array (LCA device) implements a multiplexer and counter used in time division multiplexing. With the device's flexible I/O pins, the multiplexer and counter are implemented into a single CMOS device. This application note covers time division multiplexing and the design of a multiplexer and counter. Additional information on how to design with an LCA device can be found in the LCA design methodology chapter in Monolithic Memories' LCA Design and Applications Handbook. For programming the LCA device, refer to "Configuring the LCA Device", Monolithic Memories' Application Note 182.

Principles of Multiplexing

Multiplexing efficiently utilizes data communication lines by combining multiple low-speed signals into a single, high-speed line. The two methods of performing multiplexing are Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM). TDM divides the transmission bandwidth into equal time slots where each input signal is assigned one time slot per time cycle. Once assigned, that time slot is not used by any other input. Figure 1 shows a multiplexer combining three low-speed signals into one high-speed line. Terminal A transmits during the first time slot, B transmits during the second time slot, and C transmits during the third time slot. This sequence is repeated every time cycle. FDM, on the other hand, divides the frequency spectrum among logical channels where each channel has full bandwidth of its assigned frequencies. Analog systems usually use FDM.

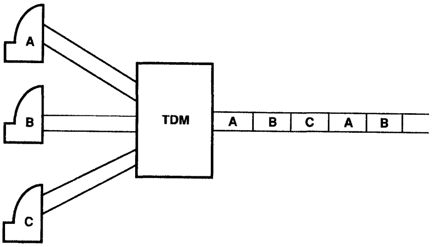


Figure 1. Time Division Multiplexing

There are many classes of TDMs including bit interleave, character interleave, statistical TDM, and T1 multiplexers. In bit interleaved TDM, each input is assigned to one time slot. Each time slot's length is one bit. Character interleaved TDM is similar to bit interleave, except that each time slot represents one character. Statistical multiplexing assigns the bandwidth into unequal slots where only the active incoming lines are assigned slots. The slots are of variable length, so each input is assigned the amount of bandwidth needed. The T1 standard specifies twenty-four 64 kbps channels multiplexed on a 1.544 Mbps high-speed link.

Data Selection/Time Slot Assignment

To transmit data from terminal A in the first time slot as shown in Figure 1, data buffering, rate adaption and data selection must be performed. Data buffering and rate adaption are required since the rates of the lines to be multiplexed are slower than the high-speed link. A serializing FIFO or shift register can implement the required buffering. Rate adaption is performed by a clock/shift scheme tailored to the exact application. Once buffered, the data must be selected in the proper order. Several methods can be used for data selection; one method is a 32-to-1 multiplexer. To implement sequential selection of input lines, an on-board counter selects the multiplexer's output. However, for random selection needed in statistical multiplexing, more flexibility is needed. A 2-to-1 multiplexer provides this flexibility by selecting between the counter and random selection inputs.

Comparison of Design Methodologies

The 32-to-1 multiplexer and counter design can be implemented in several ways. Briefly, we will evaluate discrete logic, PAL and LCA device implementations.

For this design, thirty-eight inputs, five registers, and one output are required. Conventional PLD devices do not meet this large I/O requirement. A single PLD device is not suited for this specific application because flexible I/O and buried registers are not supported. Instead, the design must be divided into four sections. The counter fits in a PALC16R6Z device and the 32-to-1 multiplexer requires three PALC20L8Z devices.

Since a large 32-to-1 multiplexer is not available as a discrete part, it must be built using smaller multiplexers. The total chip count, using discrete logic, is five devices: two 74AS850s, two 74ALS257s and one 74AS867. A combination of PAL devices and discrete logic devices is possible, however, it is also a multiple chip solution.

Monolithic Memories' LCA device is a high-density programmable CMOS circuit with flexible I/Os. It has forty pins which are user-defined as either inputs, outputs, or bidirectional. The LCA device meets the I/O requirement for this design. Since the LCA circuit allows multiple logic levels, implementing the counter does not use up output pins. Moreover an LCA device enables the 32-to-1 multiplexer with counter to be implemented in a single low-power device.

Detailed LCA Design

The M2064 LCA device in a 48-pin DIP is used in this design, although both PLCC and PGA packages are available. Eight of the forty-eight pins are reserved for programming, power, and ground, and the remaining forty I/O pins are available to the user. Table 1 shows the efficient use of the package pinout.

PIN	DESCRIPTION	TOTAL PINS
ADDR(4:0)	External Address Inputs	5 inputs
CK	External Clock Input	1 input
LD	Counter Load Input	1 input
D(31:0)	32-to-1 Multiplexer Data Inputs.	32 inputs
OUT	Output	1 output
TOTAL I/O		40 pins

Table 1. 32-to-1 Multiplexer Pins

Figure 2 shows the block diagram of the 32-to-1 multiplexer and the counter. The select lines of the 32-to-1 multiplexer are either the address lines or the output of the counter. When LD is deasserted, the 5-bit counter sequentially selects each input. Asserting the LD signal loads the counter with the external addresses.It also selects the external address for the 32-to-1 multiplexer's select lines. This permits random input selection from the external address lines which supports statistical multiplexing.

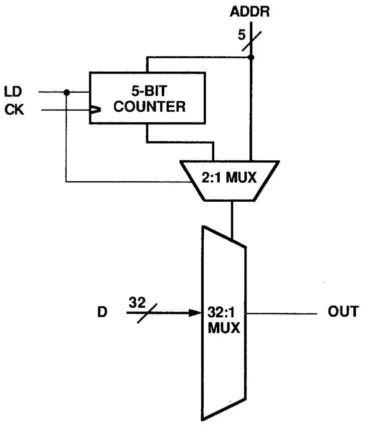


Figure 2. Block Diagram

The select signals for the 32-to-1 multiplexer are determined by the LD signal. When LD = 0, the counter selects the multiplexer's output. When LD = 1, the counter is loaded and the external address selects the output. Also, the LD asynchronously loads the 5-bit counter. Table 2 summarizes the device operation.

INPUTS		OUTPUT	FUNCTION
LD	ADDR(4:0)	OUT	
0	XXXXX	D(CNT)	Counter as Select
1	ADDR(4:0)	D(ADDR)	Address as Select and Load Counter

Table 2. 32-to-1 Multiplexer Function Table

The design was entered with FutureNet's DASH schematic capture package using high-level macros. Then the schematic capture file was translated into an LCA design file. The translation software, called PIN2LCA, partitions the design into CLBs and generates the equations for the CLBs' configuration. The translation software produces an unrouted LCA design file. This design file requires final placement of CLBs and interconnect routing which was performed using Monolithic Memories' APR software, an automatic place and route program. After the design is placed and routed, it was optimized with Monolithic Memories' XACT Design Editor System. The XACT system provides a graphic interface to manually optimize the CLB logic functions and connections.

The logic functions and interconnects of an LCA device are established with CMOS memory cells, so the array is never physically altered, although it is physically programmed. A programmable LCA device can be reconfigured for the prototype. In addition, it can be reprogrammed any number of times in the target system. With reconfigurable devices such as this one, the array can also be programmed on power-up, or whenever the design details need to be changed. For volume production, a hard-array version will be available.

Entering the Design with Futurenet

The schematic entered in FutureNet's DASH is comprised of different components called from the macro library supplied by Monolithic Memories. The M8-1 and M4-1 macros are the multiplexers which implement the 32-to-1 multiplexer as shown in Figure 3. The 5-bit counter was built from INV, C16BPRD, XOR2, and FDM macros. Since the C16BPRD macro is only a 4-bit parallel-load binary counter, the XOR2 and FDM macros were added to form another bit which increased the length to five bits.The GMUX macro, a 2-to-1 multiplexer, selects between the counter and the external address lines.

To translate from FutureNet to LCA, the PIN2LCA program is invoked. This program generates an unrouted <filename>.LCA from the FutureNet's <filename>.PIN. The PIN2LCA software partitions the design into CLBs and generates each CLB's configuration.

The PIN2LCA performs logic reduction by eliminating unused logic to maximize CLB utilization. With this design, for example, C16BPRD's reset logic is not used, and PIN2LCA eliminates all the reset logic in the C16BPRD macro. Another means to maximize CLB utilization is by combining macro logic. At times, this combined logic results in a single CLB which implements logic from two different macros. For designs not utilizing all the CLBs, this logic reduction is not required and may actually hinder placement and delay tradeoffs. If it does produce delay problems, it is possible to edit the CLBs in XACT.

At this stage in the design flow, the PIN2LCA software can generate a P-SILOS simulation file. Since the <filename>.SIM was generated from an unrouted LCA design file, the simulation only has unit delays and does not have actual routing delays. The simulation verifies the logic, but does not give any timing information. It may be necessary to make changes in DASH™ depending on the simulation results.

Place and Route Using APR

Once the logic is finalized, the LCA design must be placed and routed. Monolithic Memories' APR software performs the final placement of CLBs and interconnect routing. The input to APR is an unrouted <filename>.LCA, and a routed LCA design file and report document are the outputs.

At first glance, the APR program seems cumbersome to apply to every application. However, it can be used to create an efficient design. While the software uses random placement, it does allow tailoring for specific requirements.

Initially, the multiplexer and counter design did not route completely. To achieve good placement and 100% routing completion, several things were tried. Some worked well while others did not. An overview of how 100% routing completion was achieved is explained below.

Since this design utilizes only 53% of the LCA device, delay is a larger concern than CLB utilization. In this case, placement is critical to achieve an efficient design with the desired delays. The ideal placement would be for the 5-bit counter and 2-to-1 multiplexer to be placed near the ADDR(4:0), CK, and LD signals. The four M8-1 macros should be near the 8-bit cluster of inputs. The remaining M4-1 macro should be placed near the OUT signal.

In FutureNet, the I/Os are assigned pin numbers so that the signals are clustered into groups. These I/O assignments restrict the APR's placement of the signals and are found in the <filename>.SCP. Other restrictions and options are specified when invoking the APR software. The options, "-a", "-g", "-e", and "-k", tailor the placement for this specific design.

The APR's "-a" option specifies the depth of CLB logic levels which are considered connected. In this design, "-a2" was selected because the C16BPRD and M8-1 macros are implemented in two CLB logic levels.

The "-g" and "-e" options are related. The "-g" option specifies the number of CLBs that should be grouped together. The "-e" option determines the number of CLBs in a group which should be evaluated for all possible placements. Since the C16BPRD and M8-1 macros contain four to six CLBs, the options, "-e4" and "-g4", were used. This particular value was selected because it was the largest possible value which maintained the e>=g relationship. Avoiding values where e<g insures as much as possible that the CLBs within each group are placed in the best possible arrangement. This maintains the groups' integrity.

The "-k" option specifies the number of shapes per group. By trial and error, "-k6" works well for this design. With "-k6", six of the best arrangements or shapes were saved for each group of CLBs. Every combination of shaped group is tried with all six other grouped shapes and evaluated for best placement. Generally, the larger values result in better placements, however, run-time grows exponentially. While larger values of "-k" were tried, they did not improve the placement significantly.

Running APR with these options and restrictions did not result in a 100% routed design. Therefore, more application-specific information was required. Noting that the APR software was not fully utilizing the high-level macro information, placement could still be optimized. The APR's report file shows that the CLBs which comprise C16BPRD or the M8-1 macros were not grouped together. Instead, the CLBs were randomly spread over the device. In a constraint file, <filename>.CST, CLBs from the high-level macros could be grouped together (see Appendix A). If necessary, the exact CLB placement could be specified in the same file.

To generate the constraint file, CLBs which implement each high-level macro were identified using the cross-reference file generated by PIN2LCA, <filename>.CRF. The <filename>.CRF file is organized into three cross-reference sections: macros, CLBs, and IOBs.

The macro cross-reference section identifies and assigns a hierarchical symbol number to each macro. The symbol number is used to generate the default CLB names.

```
                            21-1, \M8-1.DWG
                               Path:
ASSIGNED
SYMBOL NUMBER              \USER\THERESA\TDM.DWG(32)
                             Title: M8-1
```

The CLB cross-reference section of the <filename>.CRF contains the CLB names. For macros with multiple CLBs, the system assigns default names. The default CLB name consists of two parts. The first is the macro symbol number as specified in the macro section and the second number is the signal name. By grouping CLBs with related symbol numbers, all the CLBs for a given macro will be placed together.

```
CLB BD Name = '21-D03':
   F = signal '21-D03', contains:
      symbol '26-OR(2)', output signal = '21-D03'
      symbol '26-AND(3)', output signal = '26-S0'
      symbol '27-OR(2)', output signal = '21-D01'
      symbol '26-AND)1)', output signal = '26-S1'
```

The IOB cross-reference section shows the IOB assignments. For this design, the assignments are those specified in the FutureNet's schematic.

```
IOB P30: Name = 'D13', Symbol = 'PIN(40)'
          I = signal 'D13'
```

Optimizing the Design with Monolithic Memories' XACT Design Editor System

Monolithic Memories' XACT Design Editor System can increase resource utilization and performance by manually modifying the placement and routing. The XACT system provides a graphic interface to specify the CLB design. All CLB logic functions and connections can be optimized for the designs' needs. With XACT, the designer can partition the design and optimize the placement of logic blocks to gain the utilization and performance required.

The clock line routing was optimized to reduce clock skew. Using the clock buffer resources, the common clock is driven by the global clock buffer. To optimize internal routing, "long lines" were used for signals with large fanouts such as the select lines for the multiplexers and the LD signal. Use of the "long lines" minimizes the delay for these control signals. The "long lines" were selected by manually routing the signal net through the programmable interconnect points (PIPs).

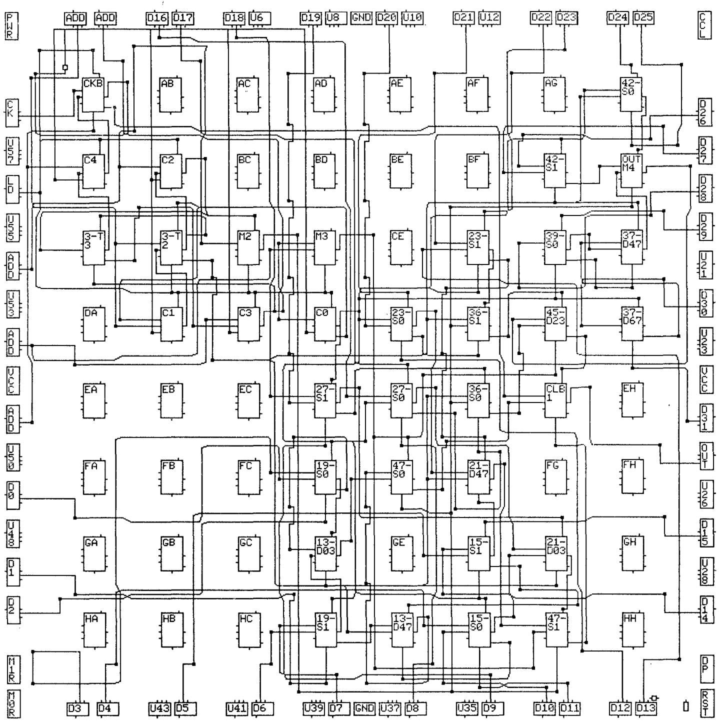
On the device itself, implementing the 5-bit counter and 2-to-1 multiplexers requires ten CLBs; and the 32-to-1 multiplexer requires twenty-four CLBs, giving a total of thirty-four CLBs. This design utilizes 53% of the LCA device.

Summary and More Information

For time division multiplexing systems, data selection can be implemented in a single CMOS device. FutureNet's DASH schematic capture with Monolithic Memories' LCA macro library was used to implement a design in an LCA device. Placement of CLBs and signal routing were performed by Monolithic Memories' APR software. Optimization, which may be required to improve performance, can be done using Monolithic Memories' XACT Design Editor System. This is useful for placing and routing critical signal nets such as clock or control signals. More information can be found in P-SILOS, FutureNet, and LCA user guides and handbooks.

This design, developed with an LCA device, is available upon request. The FutureNet drawing, LCA design and bit pattern files can be provided for programming the LCA device in an EPROM. Please ask for design XDES07.LCA.

Print World: THERESA.LCA (2064PD48-70), XACT 1.30, 10:15:46 JUL 31, 1987



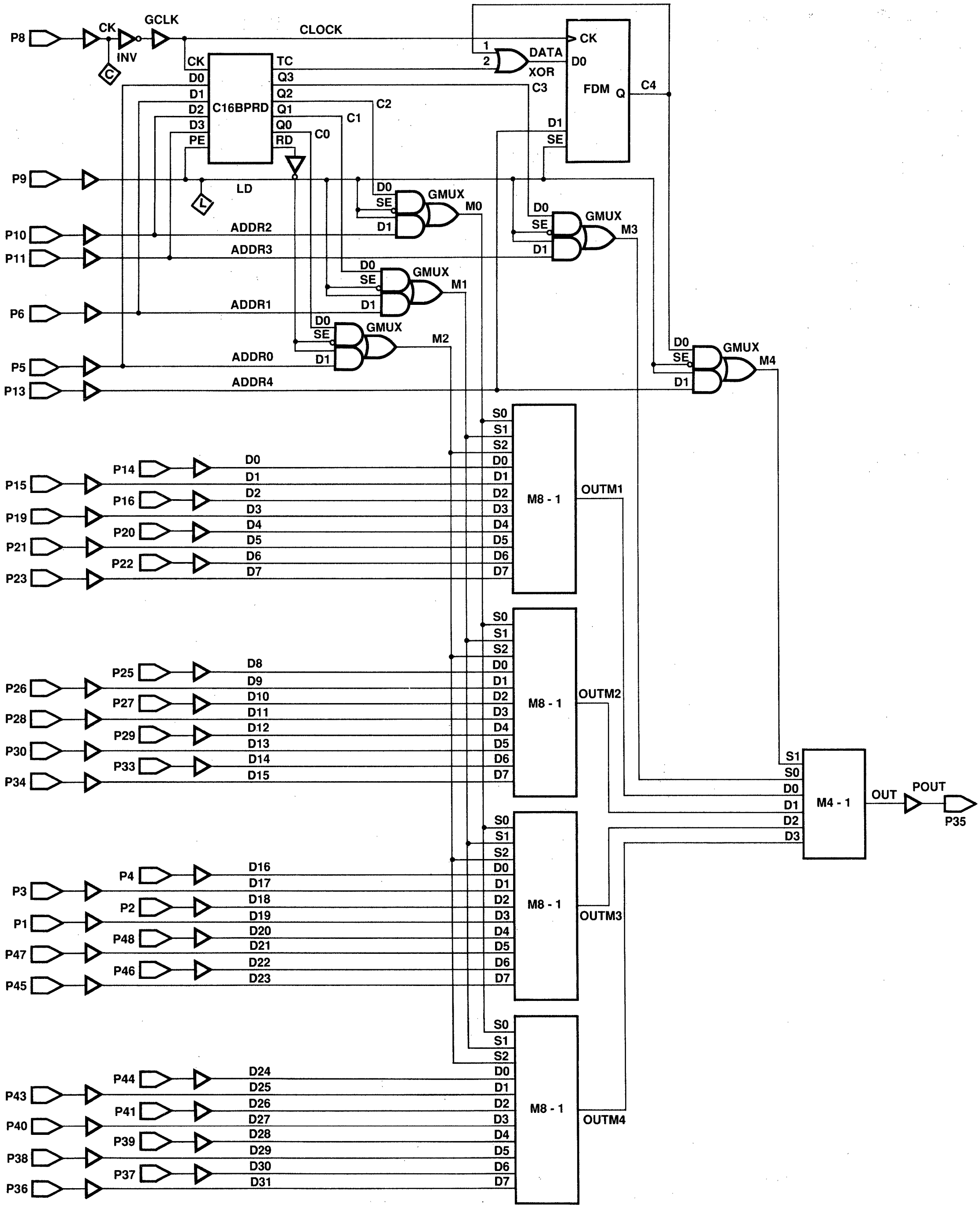


Figure 3. FutureNet Schematic

Appendix A> Constraint File

```
; CONSTRAINT FILE GROUPING MACROS IN BLOCKS

DEFINE GROUP CNT_GROUP
        3-T3 3-T2 CKB
        C4 C2 C3 C1 C0
        M3 M2;

DEFINE GROUP OUT_GROUP
        CLB1 45-D23;

DEFINE GROUP M1_GROUP
        47-S0
        19-S0 19-S1
        13-D03 13-D47;

DEFINE GROUP M2_GROUP
        47-S1
        15-S0 15-S1
        21-D03 21-D47;

DEFINE GROUP M3_GROUP
        27-S0 27-S1
        23-S0 23-S1
        36-S0 36-S1;

DEFINE GROUP M4_GROUP
        OUTM4
        42-S0 42-S1
        37-D47 37-D67
        39-S0;
```

```
$
$ Simulation file for design 'TDM.sim', type '2064N48-50'
$ Created by PIN2LCA Ver. 1.01x at 10:58:00 JUL 14, 1987
$
!INPUT TDM.sim

GLOBALRESET-      .CLK 0 S0 1 S1 $ Initial pulse to reset latches
CK.PAD            .CLK 0 0 500 1 1000 0 .REP 0
LD.PAD            .CLK 0 0 40000 1
ADDR0.PAD         .CLK 0 0 40000 0 41000 1 42000 0 .REP 40000
+ 78000 0
ADDR1.PAD         .CLK 0 0 40000 0 42000 1 44000 0 .REP 40000
+ 78000 1
ADDR2.PAD         .CLK 0 0 40000 0 44000 1 48000 0 .REP 40000
+ 78000 1
ADDR3.PAD         .CLK 0 0 40000 0 48000 1 56000 0 .REP 40000
+ 78000 0
ADDR4.PAD         .CLK 0 0 40000 0 56000 1 72000 0 .REP 40000
+ 78000 1
D0.PAD            .CLK 0 1  1000 0         40000 0 41000 1
D1.PAD            .CLK 0 0  1000 1  2000 0 40000 1 41000 0 42000 1
D2.PAD            .CLK 0 0  2000 1  3000 0 40000 1 42000 0 43000 1
D3.PAD            .CLK 0 0  3000 1  4000 0 40000 1 43000 0 44000 1
D4.PAD            .CLK 0 0  4000 1  5000 0 40000 1 44000 0 45000 1
D5.PAD            .CLK 0 0  5000 1  6000 0 40000 1 45000 0 46000 1
D6.PAD            .CLK 0 0  6000 1  7000 0 40000 1 46000 0 47000 1
D7.PAD            .CLK 0 0  7000 1  8000 0 40000 1 47000 0 48000 1
D8.PAD            .CLK 0 0  8000 1  9000 0 40000 1 48000 0 49000 1
D9.PAD            .CLK 0 0  9000 1 10000 0 40000 1 49000 0 50000 1
D10.PAD           .CLK 0 0 10000 1 11000 0 40000 1 50000 0 51000 1
D11.PAD           .CLK 0 0 11000 1 12000 0 40000 1 51000 0 52000 1
D12.PAD           .CLK 0 0 12000 1 13000 0 40000 1 52000 0 53000 1
D13.PAD           .CLK 0 0 13000 1 14000 0 40000 1 53000 0 54000 1
D14.PAD           .CLK 0 0 14000 1 15000 0 40000 1 54000 0 55000 1
D15.PAD           .CLK 0 0 15000 1 16000 0 40000 1 55000 0 56000 1
D16.PAD           .CLK 0 0 16000 1 17000 0 40000 1 56000 0 57000 1
D17.PAD           .CLK 0 0 17000 1 18000 0 40000 1 57000 0 58000 1
D18.PAD           .CLK 0 0 18000 1 19000 0 40000 1 58000 0 59000 1
D19.PAD           .CLK 0 0 19000 1 20000 0 40000 1 59000 0 60000 1
D20.PAD           .CLK 0 0 20000 1 21000 0 40000 1 60000 0 61000 1
D21.PAD           .CLK 0 0 21000 1 22000 0 40000 1 61000 0 62000 1
D22.PAD           .CLK 0 0 22000 1 23000 0 40000 1 62000 0 63000 1 73000 0
D23.PAD           .CLK 0 0 23000 1 24000 0 40000 1 63000 0 64000 1
D24.PAD           .CLK 0 0 24000 1 25000 0 40000 1 64000 0 65000 1
D25.PAD           .CLK 0 0 25000 1 26000 0 40000 1 65000 0 66000 1
D26.PAD           .CLK 0 0 26000 1 27000 0 40000 1 66000 0 67000 1
D27.PAD           .CLK 0 0 27000 1 28000 0 40000 1 67000 0 68000 1
D28.PAD           .CLK 0 0 28000 1 29000 0 40000 1 68000 0 69000 1
D29.PAD           .CLK 0 0 29000 1 30000 0 40000 1 69000 0 70000 1
D30.PAD           .CLK 0 0 30000 1 31000 0 40000 1 70000 0 71000 1
D31.PAD           .CLK 0 0 31000 1 32000 0 40000 1 71000 0 72000 1

 .MONITOR CK.PAD LD.PAD ; ADDR4.PAD ADDR3.PAD ADDR2.PAD ADDR1.PAD ADDR0.PAD ; ;
+ C4 C3 C2 C1 C0 ;
+ D0.PAD D1.PAD D2.PAD D3.PAD D4.PAD D5.PAD D6.PAD D7.PAD ;
+ D8.PAD D9.PAD D10.PAD D11.PAD D12.PAD D13.PAD D14.PAD D15.PAD ;
+ D16.PAD D17.PAD D18.PAD D19.PAD D20.PAD D21.PAD D22.PAD D23.PAD ;
+ D24.PAD D25.PAD D26.PAD D27.PAD D28.PAD D29.PAD D30.PAD D31.PAD ; OUT.PAD

 .TABLE CK.PAD LD.PAD ; ADDR4.PAD ADDR3.PAD ADDR2.PAD ADDR1.PAD ADDR0.PAD ; ;
+ C4 C3 C2 C1 C0 ;
+ D0.PAD D1.PAD D2.PAD D3.PAD D4.PAD D5.PAD D6.PAD D7.PAD ;
+ D8.PAD D9.PAD D10.PAD D11.PAD D12.PAD D13.PAD D14.PAD D15.PAD ;
+ D16.PAD D17.PAD D18.PAD D19.PAD D20.PAD D21.PAD D22.PAD D23.PAD ;
+ D24.PAD D25.PAD D26.PAD D27.PAD D28.PAD D29.PAD D30.PAD D31.PAD ; OUT.PAD
```

```
*  P - S I L O S   1U.3  *   OUTPUTS     10:26:22     07-23-87

             CL AAAAA  CCCCC DDDDDDDD DDDDDDDD DDDDDDDD DDDDDDDD O
             KD DDDDD  43210 01234567 89111111 11112222 22222233 U
             .. DDDDD        ........ ..012345 67890123 45678901 T
             PP RRRRR        PPPPPPPP PP...... ........ ........ .
             AA 43210        AAAAAAAA AAPPPPPP PPPPPPPP PPPPPPPP P
             DD .....        DDDDDDDD DDAAAAAA AAAAAAAA AAAAAAAA A
                PPPPP                   DDDDDD DDDDDDDD DDDDDDDD D
                AAAAA
                DDDDD
        TIME
           0  00 00000  00000 10000000 00000000 00000000 00000000 1
         500  10 00000  00000 10000000 00000000 00000000 00000000 1
        1000  00 00000  00000 01000000 00000000 00000000 00000000 1
        1500  10 00000  00001 01000000 00000000 00000000 00000000 1
        2000  00 00000  00001 00100000 00000000 00000000 00000000 1
        2500  10 00000  00010 00100000 00000000 00000000 00000000 1
        3000  00 00000  00010 00010000 00000000 00000000 00000000 1
        3500  10 00000  00011 00010000 00000000 00000000 00000000 1
        4000  00 00000  00011 00001000 00000000 00000000 00000000 1
        4500  10 00000  00100 00001000 00000000 00000000 00000000 1
        5000  00 00000  00100 00000100 00000000 00000000 00000000 1
        5500  10 00000  00101 00000100 00000000 00000000 00000000 1
        6000  00 00000  00101 00000010 00000000 00000000 00000000 1
        6500  10 00000  00110 00000010 00000000 00000000 00000000 1
        7000  00 00000  00110 00000001 00000000 00000000 00000000 1
        7500  10 00000  00111 00000001 00000000 00000000 00000000 1
        8000  00 00000  00111 00000000 10000000 00000000 00000000 1
        8500  10 00000  01000 00000000 10000000 00000000 00000000 1
        9000  00 00000  01000 00000000 01000000 00000000 00000000 1
        9500  10 00000  01001 00000000 01000000 00000000 00000000 1
       10000  00 00000  01001 00000000 00100000 00000000 00000000 1
       10500  10 00000  01010 00000000 00100000 00000000 00000000 1
       11000  00 00000  01010 00000000 00010000 00000000 00000000 1
       11500  10 00000  01011 00000000 00010000 00000000 00000000 1
       12000  00 00000  01011 00000000 00001000 00000000 00000000 1
       12500  10 00000  01100 00000000 00001000 00000000 00000000 1
       13000  00 00000  01100 00000000 00000100 00000000 00000000 1
       13500  10 00000  01101 00000000 00000100 00000000 00000000 1
       14000  00 00000  01101 00000000 00000010 00000000 00000000 1
       14500  10 00000  01110 00000000 00000010 00000000 00000000 1
       15000  00 00000  01110 00000000 00000001 00000000 00000000 1
       15500  10 00000  01111 00000000 00000001 00000000 00000000 1
       16000  00 00000  01111 00000000 00000000 10000000 00000000 1
       16500  10 00000  10000 00000000 00000000 10000000 00000000 1
       17000  00 00000  10000 00000000 00000000 01000000 00000000 1
       17500  10 00000  10001 00000000 00000000 01000000 00000000 1
       18000  00 00000  10001 00000000 00000000 00100000 00000000 1
       18500  10 00000  10010 00000000 00000000 00100000 00000000 1
       19000  00 00000  10010 00000000 00000000 00010000 00000000 1
       19500  10 00000  10011 00000000 00000000 00010000 00000000 1
       20000  00 00000  10011 00000000 00000000 00001000 00000000 1
       20500  00 00000  10011 00000000 00000000 00001000 00000000 1
       21000  00 00000  10100 00000000 00000000 00000100 00000000 1
       21500  10 00000  10101 00000000 00000000 00000100 00000000 1
       22000  00 00000  10101 00000000 00000000 00000010 00000000 1
       22500  10 00000  10110 00000000 00000000 00000010 00000000 1
       23000  00 00000  10110 00000000 00000000 00000001 00000000 1
       23500  10 00000  10111 00000000 00000000 00000001 00000000 1
       24000  00 00000  10111 00000000 00000000 00000000 10000000 1
       24500  10 00000  11000 00000000 00000000 00000000 10000000 1
       25000  00 00000  11000 00000000 00000000 00000000 01000000 1
       25500  10 00000  11001 00000000 00000000 00000000 01000000 1
       26000  00 00000  11001 00000000 00000000 00000000 00100000 1
       26500  10 00000  11010 00000000 00000000 00000000 00100000 1
       27000  00 00000  11010 00000000 00000000 00000000 00010000 1
       27500  10 00000  11011 00000000 00000000 00000000 00010000 1
       28000  00 00000  11011 00000000 00000000 00000000 00001000 1
       28500  10 00000  11100 00000000 00000000 00000000 00001000 1
       29000  00 00000  11100 00000000 00000000 00000000 00000100 1
       29500  10 00000  11101 00000000 00000000 00000000 00000100 1
       30000  00 00000  11101 00000000 00000000 00000000 00000010 1
       30500  10 00000  11110 00000000 00000000 00000000 00000010 1
       31000  00 00000  11110 00000000 00000000 00000000 00000001 1
       31500  10 00000  11111 00000000 00000000 00000000 00000001 1
       32000  00 00000  11111 00000000 00000000 00000000 00000000 1
       32500  10 00000  00000 00000000 00000000 00000000 00000000 0
       33000  00 00000  00000 00000000 00000000 00000000 00000000 0
       39000  00 00000  00110 00000000 00000000 00000000 00000000 0
       39500  10 00000  00111 00000000 00000000 00000000 00000000 0
       40000  01 00000  00111 01111111 11111111 11111111 11111111 0
       40500  11 00000  01000 01111111 11111111 11111111 11111111 0
       41000  01 00001  01000 10111111 11111111 11111111 11111111 0
       41500  11 00001  00001 10111111 11111111 11111111 11111111 0
       42000  01 00010  00001 11011111 11111111 11111111 11111111 0
       42500  11 00010  00010 11011111 11111111 11111111 11111111 0
```

```
*  P - S I L O S   1U.3  *   OUTPUTS     10:26:22     07-23-87

             CL AAAAA  CCCCC DDDDDDDD DDDDDDDD DDDDDDDD DDDDDDDD O
             KD DDDDD  43210 01234567 89111111 11112222 22222233 U
             .. DDDDD        ........ ..012345 67890123 45678901 T
             PP RRRRR        PPPPPPPP PP...... ........ ........ .
             AA 43210        AAAAAAAA AAPPPPPP PPPPPPPP PPPPPPPP P
             DD .....        DDDDDDDD DDAAAAAA AAAAAAAA AAAAAAAA A
                PPPPP                   DDDDDD DDDDDDDD DDDDDDDD D
                AAAAA
                DDDDD
   TIME
  43000      01 00011  00010 11101111 11111111 11111111 11111111 0
  43500      11 00011  00011 11101111 11111111 11111111 11111111 0
  44000      01 00100  00011 11110111 11111111 11111111 11111111 0
  44500      11 00100  00000 11110111 11111111 11111111 11111111 0
  45000      01 00101  00000 11111011 11111111 11111111 11111111 0
  45500      11 00101  00101 11111011 11111111 11111111 11111111 0
  46000      01 00110  00101 11111101 11111111 11111111 11111111 0
  46500      11 00110  00110 11111101 11111111 11111111 11111111 0
  47000      01 00111  00110 11111110 11111111 11111111 11111111 0
  47500      11 00111  00111 11111110 11111111 11111111 11111111 0
  48000      01 01000  00111 11111111 01111111 11111111 11111111 0
  48500      11 01000  01100 11111111 01111111 11111111 11111111 0
  49000      01 01001  01100 11111111 10111111 11111111 11111111 0
  49500      11 01001  01001 11111111 10111111 11111111 11111111 0
  50000      01 01010  01001 11111111 11011111 11111111 11111111 0
  50500      11 01010  01010 11111111 11011111 11111111 11111111 0
  51000      01 01011  01010 11111111 11101111 11111111 11111111 0
  51500      11 01011  01011 11111111 11101111 11111111 11111111 0
  52000      01 01100  01011 11111111 11110111 11111111 11111111 0
  52500      11 01100  01000 11111111 11110111 11111111 11111111 0
  53000      01 01101  01000 11111111 11111011 11111111 11111111 0
  53500      11 01101  01101 11111111 11111011 11111111 11111111 0
  54000      01 01110  01101 11111111 11111101 11111111 11111111 0
  54500      11 01110  01110 11111111 11111101 11111111 11111111 0
  55000      01 01111  01110 11111111 11111110 11111111 11111111 0
  55500      11 01111  01111 11111111 11111110 11111111 11111111 0
  56000      01 10000  01111 11111111 11111111 01111111 11111111 0
  56500      11 10000  10100 11111111 11111111 01111111 11111111 0
  57000      01 10001  10100 11111111 11111111 10111111 11111111 0
  57500      11 10001  10001 11111111 11111111 10111111 11111111 0
  58000      01 10010  10001 11111111 11111111 11011111 11111111 0
  58500      11 10010  10010 11111111 11111111 11011111 11111111 0
  59000      01 10011  10010 11111111 11111111 11101111 11111111 0
  59500      11 10011  10011 11111111 11111111 11101111 11111111 0
  60000      01 10100  10011 11111111 11111111 11110111 11111111 0
  60500      11 10100  10000 11111111 11111111 11110111 11111111 0
  61000      01 10101  10000 11111111 11111111 11111011 11111111 0
  61500      11 10101  10101 11111111 11111111 11111011 11111111 0
  62000      01 10110  10101 11111111 11111111 11111101 11111111 0
  62500      11 10110  10110 11111111 11111111 11111101 11111111 0
  63000      01 10111  10110 11111111 11111111 11111110 11111111 0
  63500      11 10111  10111 11111111 11111111 11111110 11111111 0
  64000      01 11000  10111 11111111 11111111 11111111 01111111 0
  64500      11 11000  11100 11111111 11111111 11111111 01111111 0
  65000      01 11001  11100 11111111 11111111 11111111 10111111 0
  65500      11 11001  11001 11111111 11111111 11111111 10111111 0
  66000      01 11010  11001 11111111 11111111 11111111 11011111 0
  66500      11 11010  11010 11111111 11111111 11111111 11011111 0
  67000      01 11011  11010 11111111 11111111 11111111 11101111 0
  67500      11 11011  11011 11111111 11111111 11111111 11101111 0
  68000      01 11100  11011 11111111 11111111 11111111 11110111 0
  68500      11 11100  11000 11111111 11111111 11111111 11110111 0
  69000      01 11101  11000 11111111 11111111 11111111 11111011 0
  69500      11 11101  11101 11111111 11111111 11111111 11111011 0
  70000      01 11110  11101 11111111 11111111 11111111 11111101 0
  70500      11 11110  11110 11111111 11111111 11111111 11111101 0
  71000      01 11111  11110 11111111 11111111 11111111 11111110 0
  71500      11 11111  11111 11111111 11111111 11111111 11111110 0
  72000      01 00000  11111 11111111 11111111 11111111 11111111 0
  72500      11 00000  00100 11111111 11111111 11111111 11111111 1
  73000      01 00001  00100 11111111 11111111 11111101 11111111 1
  73500      11 00001  00001 11111111 11111111 11111101 11111111 1
  74000      01 00010  00001 11111111 11111111 11111101 11111111 1
  74500      11 00010  00010 11111111 11111111 11111101 11111111 1
  75000      01 00011  00010 11111111 11111111 11111101 11111111 1
  75500      11 00011  00011 11111111 11111111 11111101 11111111 1
  76000      01 00100  00011 11111111 11111111 11111101 11111111 1
  76500      11 00100  00000 11111111 11111111 11111101 11111111 1
  77000      01 00101  00000 11111111 11111111 11111101 11111111 1
  77500      11 00101  00101 11111111 11111111 11111101 11111111 1
  78000      01 10110  00101 11111111 11111111 11111101 11111111 1
  78500      11 10110  10110 11111111 11111111 11111101 11111111 0
  79000      01 10110  10110 11111111 11111111 11111101 11111111 0
  79500      11 10110  10110 11111111 11111111 11111101 11111111 0
  80000      01 10110  10110 11111111 11111111 11111101 11111111 0
```

Advanced
Micro
Devices

AN-172

Dual 32-bit Serial CRC Error Detection in an LCA Device

Karen Spesard

Abstract

The transmission and reception of digital data over local area networks (LANs) is more popular than ever. To transmit reliable information from one system to another over a LAN, an efficient error-detection technique is necessary. The error-detection scheme chosen by the IEEE for the Ethernet, a local area network developed by the Xerox Corporation, uses a 32-bit cyclic redundancy check (CRC).

A Dual 32-bit Serial CRC transmitter-receiver for the Ethernet is implemented in one Logic Cell™ Array (LCA device). The LCA device is a RAM-based CMOS circuit which is electrically programmable. It allows the designer to make design changes as necessary to accommodate other data communication protocols as well as other applications. This applications note describes the Ethernet CRC and how it is implemented in the versatile LCA device.

Dual 32-bit Serial CRC Error Detection in an LCA Device

Karen Spesard

Introduction

In transferring digital information from computer to computer, or from computer to peripheral devices, it is possible that errors in transmission may be introduced. The Cyclic Redundancy Check, or CRC, developed for the data communications industry will detect most of these errors. With one LCA device, a dual 32-bit serial CRC transmitter-receiver was implemented for the Ethernet, one of the industry's most popular local networks. This article describes the methodology used for designing the Dual 32-bit Serial CRC with an LCA device.

Overview of the CRC

Central to a serial hardware CRC system is a set of n-bit linear feedback shift registers, or LFSRs. These registers are designed to represent a fixed generator polynomial G(x). The generator polynomial is the divisor in equation 1,

$$x^n \frac{D(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}, (1)$$

and D(x) is the data polynomial, which is transmitted and checked for errors. The data polynomial is prescaled by x^n, the length of the generator polynomial, to insure the remainder is always different from the data itself.

The data is shifted in the LFSR and subsequently divided by G(x). The division generates the remainder polynomial R(x) but ignores the quotient polynomial Q(x). The remainder polynomial or redundancy check-bits remain in the registers after all of the data has been shifted in. The check-bits are then shifted out and appended to the data-bits to produce the encoded data. The encoded data becomes D(x) + R(x).

The encoded data is received in a similar LFSR with the same G(x). Because addition and subtraction operations are identical in modulo-2 arithmetic, equation (1) becomes:

$Q(x)G(x) = x^nD(x) + R(x)$. (2)

Since D(x) was prescaled, and appending the remainder to the data is equivalent to adding it, the new encoded polynomial should be exactly divisible by G(x). Thus, if the remainder of this operation is zero, all of the original transmitted data-bits are assumed unaltered. If the remainder is anything other than zero, an error occurred and an error flag is set.

Generating CRC Bits and Checking Data

An example of the LFSR hardware needed to generate CRC bits serially for a three-bit G(x) is shown in Figure 1. In this example, the output of the last register is XORed, or shift subtracted, with the data-bit before feeding back to registers X0 and X2. The feedback term positions in the LFSR correspond to all but the highest power of x in the generator polynomial. In this case, the feedback positions are determined from x^2 and x^0.

The redundancy check-bits can be calculated from Table 1 for $G(x) = x^3 + x^2 + 1$ where the internal state of each register in the LFSR after every shift is shown. Therefore, if the data is 10011, the redundancy check-bits are D5 = 0, D6 = 1, and D7 = 0 after the last data-bit is shifted in (5th shift). This is because D0 = 1, D1 = 1, D2 = 0, D3 = 0, and D4 = 1. ($D(x) = 1 + 0x + 0x^2 + 1x^3 + 1x^4$ and the least significant bit, LSB, is the first bit transmitted.) The check-bits are shifted out from the LFSR by

REGISTERS	X0	X1	X2	COMMENTS
Initialize	0	0	0	If registers initialized to 0
1st shift	D0	0	D0	
2nd shift	D0⊕D1	D0	D0⊕D1	
3rd shift	D0⊕D1⊕D2	D0⊕D1	D1⊕D2	
4th shift	D1⊕D2⊕D3	D0⊕D1⊕D2	D0⊕D2⊕D3	
5th shift	D0⊕D2⊕D3⊕D4	D1⊕D2⊕D3	D1⊕D3⊕D4	Redundancy check bits
6th shift	D1⊕D3⊕D4⊕D5	D0⊕D2⊕D3⊕D4	D2⊕D4⊕D5	
7th shift	D2⊕D4⊕D5⊕D6	D1⊕D3⊕D4⊕D5	D0⊕D3⊕D5⊕D6	
8th shift	D0⊕D3⊕D5⊕D6⊕D7	D2⊕D4⊕D5⊕D6	D0⊕D1⊕D4⊕D6⊕D7	Zero remainder

⊕ ≡ EXOR

Table 1. Calculation of Redundancy Check-Bits for $G(X) = X^3 + X^2 + 1$

disabling the feedback terms of the LFSR. The check-bits can be verified by long division as in Figure 2.

The encoded data then becomes 010 10011. To check for data integrity, the encoded data is shifted in another LFSR with the same G(x). This time the last bit (8th bit) is shifted in, and if the message was unaltered, the bits residing in the registers are zero. If the redundancy bits are not zero, an error occurred. This can also be verified by long division as in Figure 3.

In summary, CRC modifies the data polynomial so that it is exactly divisible by a fixed polynomial G(x). When the modified polynomial is received, it is checked for the exact division by G(x). If an error occurred, the RDYFLG is set.

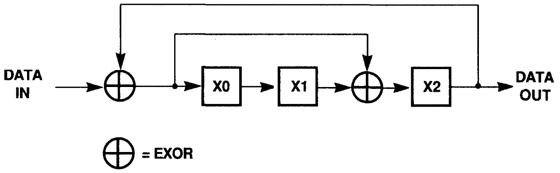


Figure 1. LFSR for G(X) = $X^3 + X^2 + 1$

```
           1001
1101 | 11001000    PRESCALED DATA
       1101        WITH 3 ZERO BITS
       ----------
          11000    REMAINDER
          11010
       ----------
            010
```

Figure 2. Long Division Example - With Pure Data

```
           1001
1101 | 11001010    ENCODED DATA
       1101
       ----------
         011010
          11010
       ----------
              0    NO REMAINDER
```

Figure 3. Long Division Example - With Encoded Data

Why Use LCA Devices in CRC?

The 32-bit serial CRC, with transmission and reception sections, can be implemented in several ways. One way would be to acquire standard CRC chips. These chips, however, only exist with specific G(x) and therefore can be inflexible. Another method would be to design the transmission and reception sections of the CRC with traditional PLD devices. This would require eight PLD devices for the dual 32-bit LFSRs (eight bits in each PLD) even without additional control logic. A third method would use an LCA device, as it would take only one to implement the dual 32-bit CRC and with more flexibility than a dedicated custom part.

In addition, the LCA device can provide designers with more control after design release than the alternative solutions. For example, the designer can change the generator polynomial or the control logic "on-the-fly" by merely reprogramming the configuration data from separate sections of an EPROM. A large EPROM, 8Kx8 or 16Kx8, can typically store three and six configuration patterns, respectively.

32-Bit CRC Design

CRC error detection is one of the best methods for checking the validity of large frames of information. It can detect all errors within n successive bits, all errors with an odd number of bits in error for even G(x), and, of course, all error patterns that are not divisible by G(x).

The IEEE-802.3/Ethernet Local Area Network Standard defines a 32-bit CRC code. This code works with data ranges from 46 to 1500 bytes, and is also used in the Autodin-II Network.

The generator polynomial used for the Ethernet 32-bit CRC is defined as:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1,$$

where the coefficients of the generator polynomial correspond to the feedback positions in the LFSR.

Figure 4 shows the block diagram of the Dual 32-bit Serial CRC. As the diagram illustrates, the transmitter portion of the CRC is controlled separately from the receiver portion.

The standard specifies that on the transmitter end, the shift registers are preloaded to ones with INITA, and CONTROLA is held HIGH while the incoming data bits are shifted in to generate the CRC. When all of the data bits have been entered, CONTROLA is held LOW and the complemented CRC is shifted out for transmission.

On the receiver end, the data bits and the complemented check bits are sent. When the end of the frame is reached, the remainder is checked. If no error occurs, the final contents of the shift register are:

```
X31                                    X0
  11000111 00000100 11011101 01111011
```

This remainder is not 0 because the check-bits are complemented before being transmitted to the receiver.

CRC Logic Implementation

Figures 5 and 6 show the complete design for the 32-bit CRC with the standard generator polynomial described above. The generator polynomial is implemented with thirty-two registers in each LFSR. The feedback terms (Q31 xor DIN) are at registers Q0, Q1, Q2, Q4, Q5, Q7, ... Q23, and Q26 and correspond to the Ethernet generator polynomial, G(x) given above.

The registers in the LFSRs are D-type flip-flops and are clocked synchronously. In the transmitter portion, two pins - CONTROL and INIT - are respectively added for initializing the registers and controlling the feedback terms. When INIT is HIGH, all registers are initialized to HIGH. When CONTROL is HIGH, the multiplexer shifts out data bits and generates check-bits. When CONTROL is LOW, the feedback term is disabled and inverted redundancy check-bits are shifted out from the LFSR.

The five-bit synchronous ripple carry counter is enabled with the control line LOW and reset with the control line HIGH. When the five count bits reach all ones, the thirty-two check bits are shifted out of the LFSR, and the RDYFLAG is set.

The receiver LFSR is essentially the same as the transmitter LFSR. However, after data and the thirty-two check bits are received, the data is checked for errors. If an error occurred, the error flag is set.

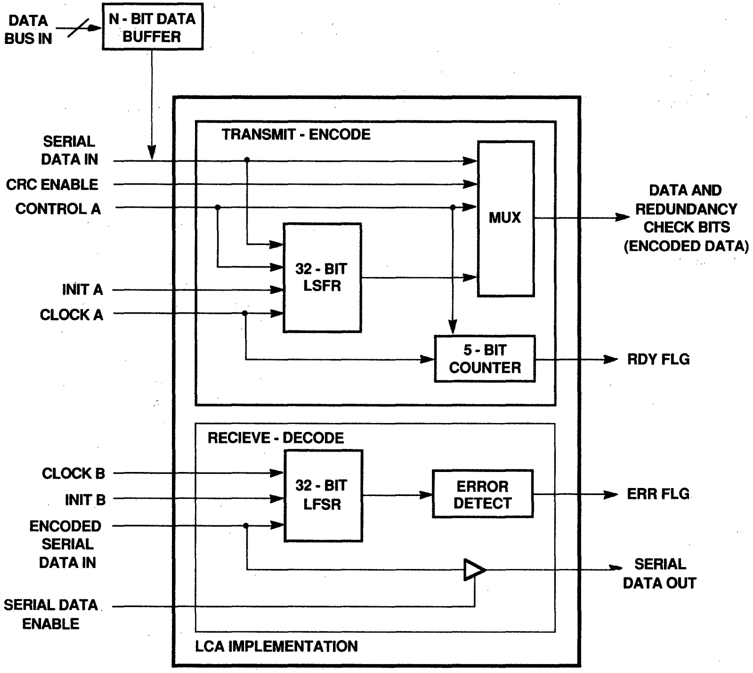


Figure 4. Block Diagram of the Dual 32-Bit Serial CRC

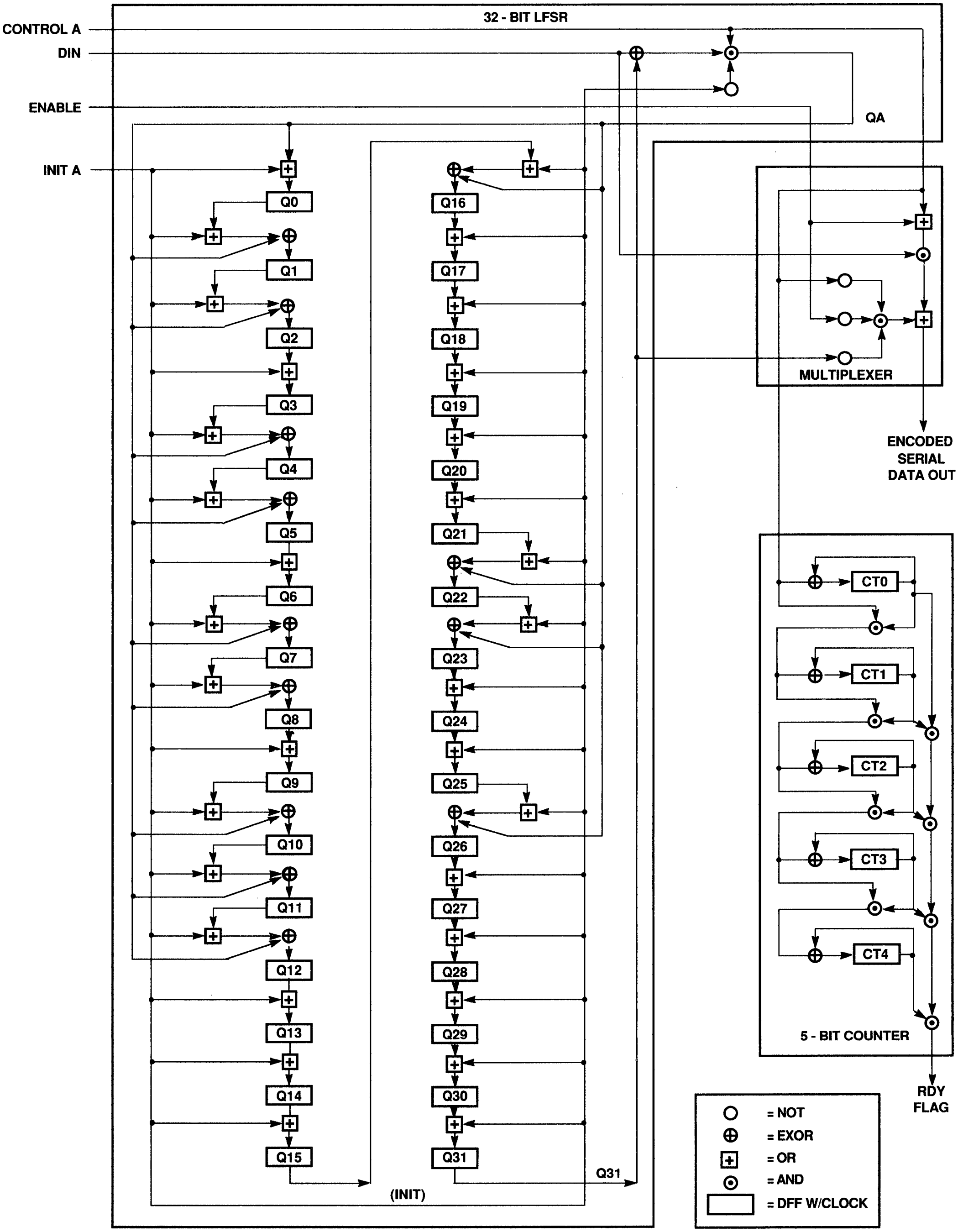


Figure 5. Logic Implementation of Transmitter-Encoder Section

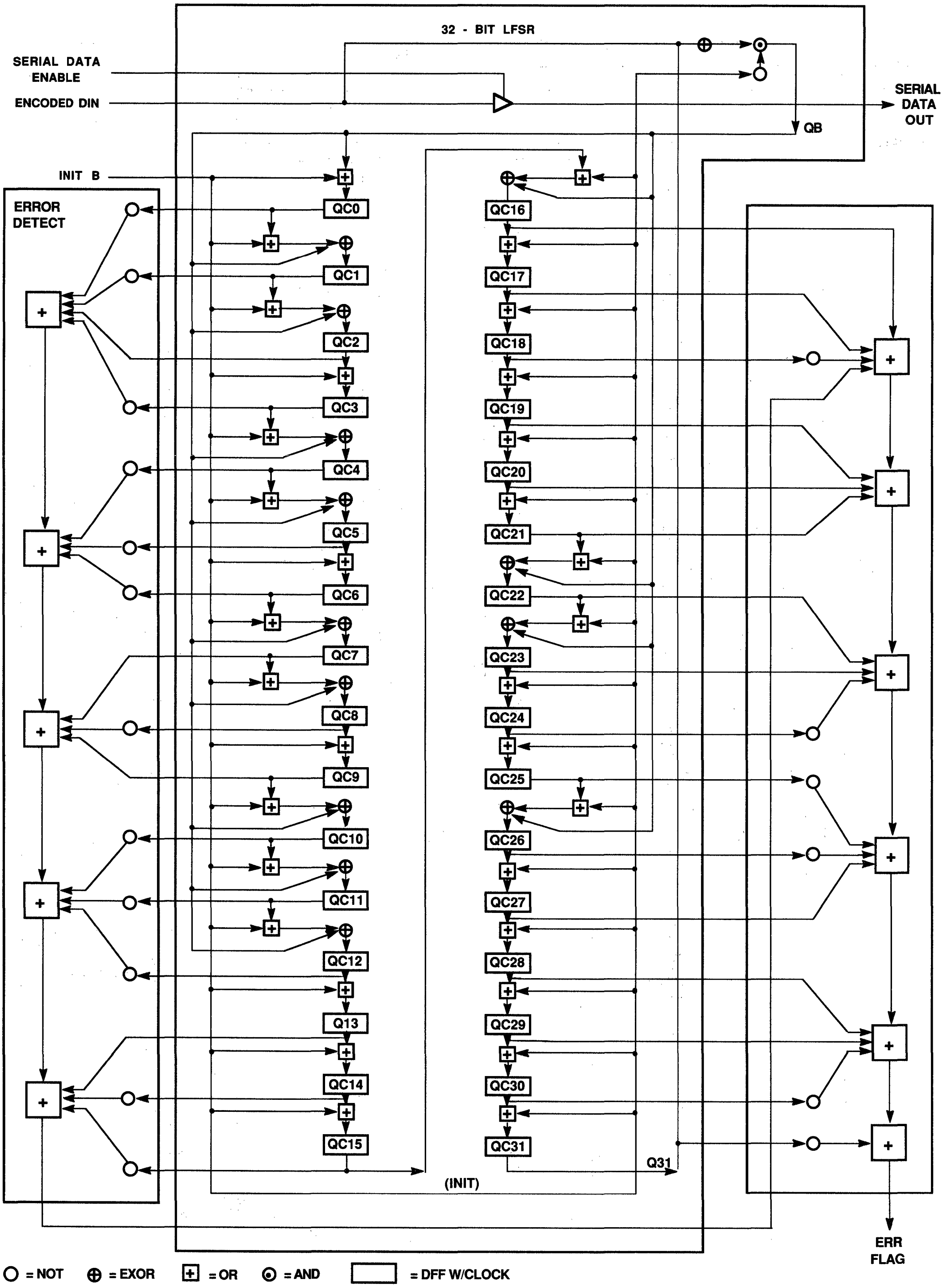


Figure 6. Logic Implementation of Receiver-Decoder Section

Logic Cell Array Implementation

The Dual 32-bit Serial CRC circuit described above is implemented in an M2018 LCA device. It contains one hundred configurable logic blocks (CLBs) and seventy-four user-configurable input/output blocks (IOBs) providing a total of 1800 usable logic gates. The logic is partitioned and placed in a graphics environment on the IBM® PC-XT™ or PC-AT™ using the Monolithic Memories' XACT™ LCA Editor.

The Monolithic Memories' XACT LCA Editor provides the tools from which the logic and interconnects (CLBs, IOBs, and programmable interconnect points) are constructed. The configuration of specified logic blocks can be made with the XACT "Edit Block" command. Once in the "Edit Block" menu, a combinational, latched, or registered equation can be selected with the mouse. The clock input polarity, set and reset functions can also be selected as required. The CLB can be configured as one output function of four Boolean input variables, or two functions of three Boolean input variables. The logic equations can be input from the PC keyboard using Boolean algebra, or from the mouse using a Karnaugh map. For an IOB, the "Edit Block" menu can be used to select pad and/or latched inputs, buffered and/or three-state control outputs, or bidirectional I/Os.

The programmable interconnect points (PIPs), are configured using the XACT "NET" and "PIN" menus. The features used from the "NET" and "PIN" menus are summarized below.

Command	Function
Name Net	Names a Net
Add Net	Names and Adds a Net
Add Pin	Adds Net and Automatically Routes a Net
Edit Net	Manually Routes a Net
Route Net	Routes Named Nets

When the nets are routed, the PIPs are programmed automatically.

An example of a CLB is shown in Figure 7. Here the output of the CLB is combinational, and the CLB is configured as a multiplexer. The multiplexer selects the serial data when CONTROL or ENABLE are HIGH and selects $\overline{Q31}$ when CONTROL and ENABLE are LOW. The Karnaugh map and the Boolean equation in the figure illustrate the logic used to implement this function. The output of the multiplexer is named ENCDATAOUT.

Figure 8 shows the configuration of the CLBs for the modulo-2 addition of the feedback signal with the data, when INIT is LOW and CONTROL is HIGH. The output again is combinational and is named QA.

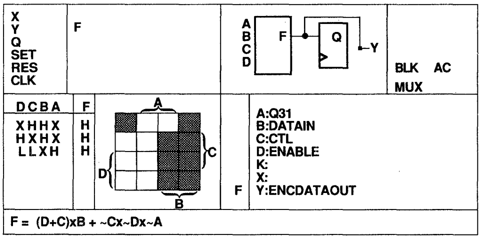


Figure 7. Mux for Data/Check Bit Select

Figures 9 to 11 show the configuration used to implement the logic for the first few shift registers, corresponding to equations in Tables 2 and 3. Note that these shift registers are configured as registered equations. Figure 12 illustrates the signal configuration of the RDYFLG in the transmitter section

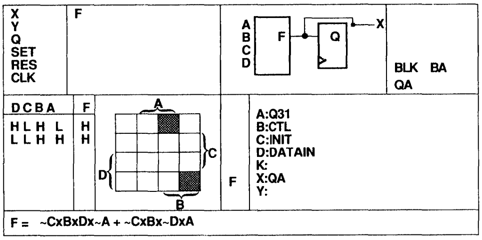


Figure 8. Modulo-2 Addition of Q31 and DATAIN

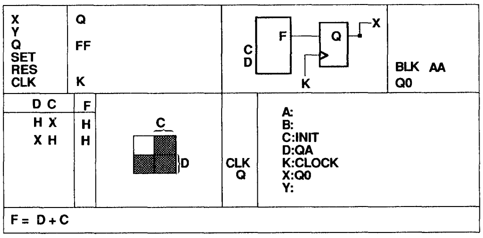


Figure 9. The 1st Bit of the LFSR

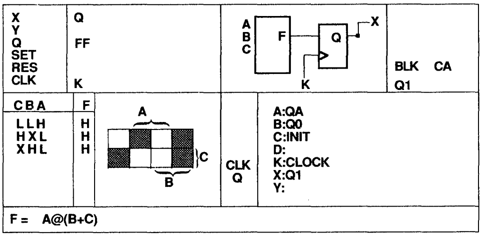


Figure 10. The 2nd Bit of the LFSR

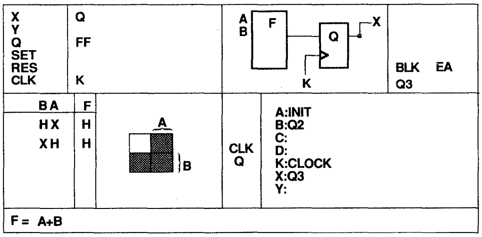


Figure 11. The 4th Bit of the LFSR

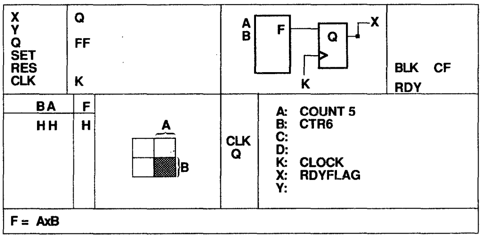


Figure 12. The RDY Flag Signal

of the LFSR. It is also configured as a registered equation. The RDYFLAG is set when COUNT5 is HIGH and CTR14 is HIGH. CTR14 is an AND gate of input bits 1 - 4.

The placed and routed Dual 32-bit CRC design requires sixty-four CLBs for the two LFSRs and two CLBs for the feedback controls. It also requires one CLB for the multiplexer, five for the counter, two for the RDYFLG and eleven for the ERRFLG. The design, therefore, requires eighty-five CLBs and utilizes 85% of the LCA device. A 68-pin package, 2018NL68, was chosen for this implementation. This 2018NL68 has a speed grade of 70 MHz, the flip-flop toggle rate. A copy of the Monolithic Memories' XACT EDITLCA file is shown in Figure 13 with all of the CLBs and IOBs labelled.

```
Print World: 32BITCRC.LCA (2018NL68-70), XACT 1.30, 16:20:22 JUN 4, 1987Print World: 32BITCRC.LCA (2018NL68-70;
```

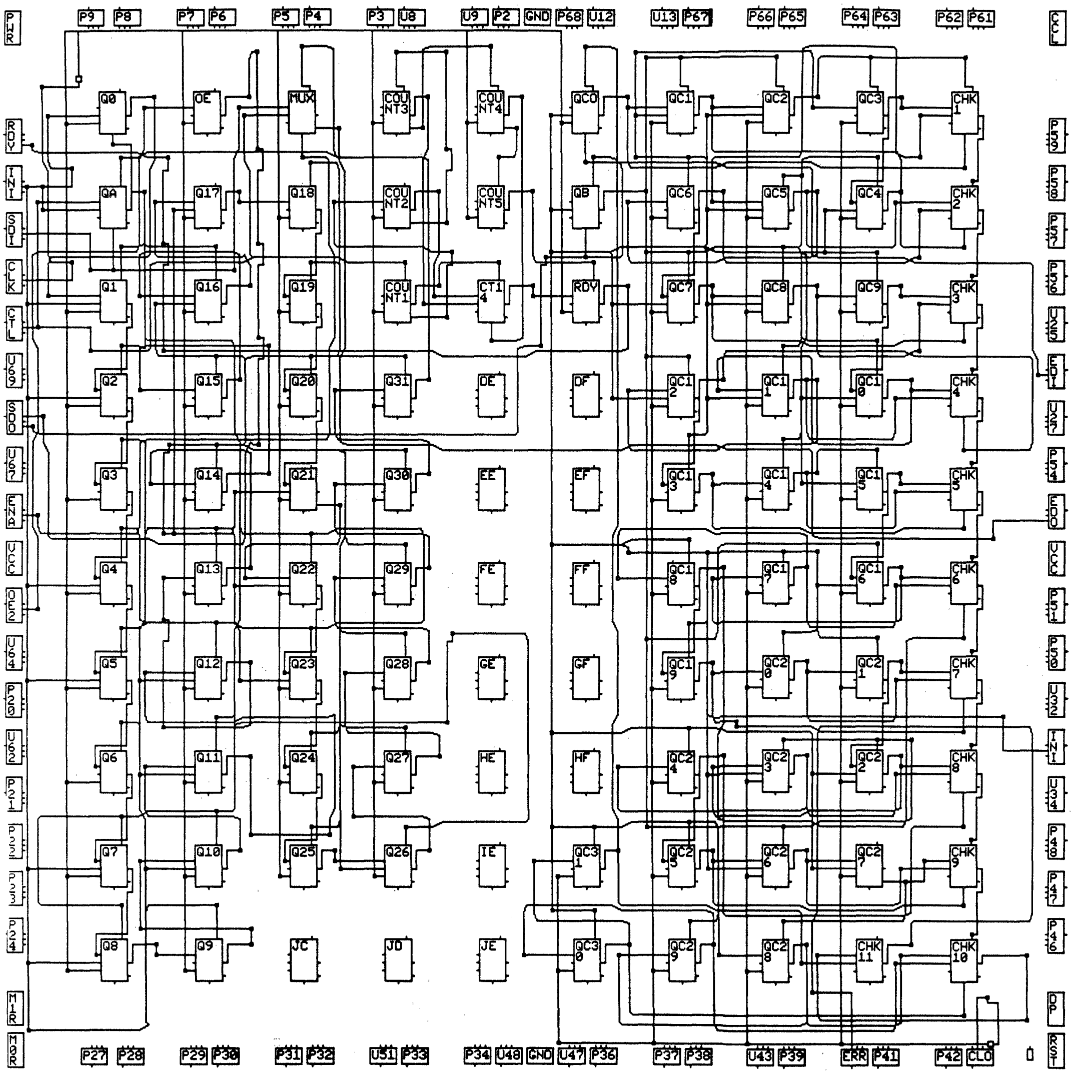


Figure 13. Monolithic Memories' XACT "EDITLCA" File for the Dual 32-Bit Serial CRC

```
                  EQUATIONS FOR 32-BIT SERIAL CRC
                       TRANSMISSION SECTION

QA   =  CTL*/INIT(Q31@DIN)      ;MOD 2 ADDITION, IF CTL*/INIT
Q0   := QA+INIT                 ;SHIFT QA OR INITIALIZE
Q1   := (Q0+INIT)@QA            ;SHIFT Q0@QA OR INITIALIZE
Q2   := (Q1+INIT)@QA            ;SHIFT Q1@QA OR INITIALIZE
Q3   := Q2+INIT                 ;SHIFT Q2 OR INITIALIZE
Q4   := (Q3+INIT)@QA            ;SHIFT Q3@QA OR INITIALIZE
Q5   := (Q4+INIT)@QA            ;SHIFT Q4@QA OR INITIALIZE
Q6   := Q5+INIT                 ;SHIFT Q5 OR INITIALIZE
Q7   := (Q6+INIT)@QA            ;SHIFT Q6@QA OR INITIALIZE
Q8   := (Q7+INIT)@QA            ;SHIFT Q7@QA OR INITIALIZE
Q9   := Q8+INIT                 ;SHIFT Q8 OR INITIALIZE
Q10  := (Q9+INIT)@QA            ;SHIFT Q9@QA OR INITIALIZE
Q11  := (Q10+INIT)@QA           ;SHIFT Q10@QA OR INITIALIZE
Q12  := (Q11+INIT)@QA           ;SHIFT Q11@QA OR INITIALIZE
Q13  := Q12+INIT                ;SHIFT Q12 OR INITIALIZE
Q14  := Q13+INIT                ;SHIFT Q13 OR INITIALIZE
Q15  := Q14+INIT                ;SHIFT Q14 OR INITIALIZE
Q16  := (Q15+INIT)@QA           ;SHIFT Q15@QA OR INITIALIZE
Q17  := Q16+INIT                ;SHIFT Q16 OR INITIALIZE
Q18  := Q17+INIT                ;SHIFT Q17 OR INITIALIZE
Q19  := Q18+INIT                ;SHIFT Q18 OR INITIALIZE
Q20  := Q19+INIT                ;SHIFT Q19 OR INITIALIZE
Q21  := Q20+INIT                ;SHIFT Q20 OR INITIALIZE
Q22  := (Q21+INIT)@QA           ;SHIFT Q21@QA OR INITIALIZE
Q23  := (Q22+INIT)@QA           ;SHIFT Q22@QA OR INITIALIZE
Q24  := Q23+INIT                ;SHIFT Q23 OR INITIALIZE
Q25  := Q24+INIT                ;SHIFT Q24 OR INITIALIZE
Q26  := (Q25+INIT)@QA           ;SHIFT Q25@QA OR INITIALIZE
Q27  := Q26+INIT                ;SHIFT Q26 OR INITIALIZE
Q28  := Q27+INIT                ;SHIFT Q27 OR INITIALIZE
Q29  := Q28+INIT                ;SHIFT Q28 OR INITIALIZE
Q30  := Q29+INIT                ;SHIFT Q29 OR INITIALIZE
Q31  := Q30+INIT                ;SHIFT Q30 OR INITIALIZE

ENCDATAOUT = (ENABLE+CTL)*SERDATAIN+(/ENABLE*/CTL*Q31)
RDYFLG := COUNT5*CTR14
```

Table 2.

```
                    EQUATIONS FOR 32-BIT SERIAL CRC
                           RECEPTION SECTION

QB    =  /INIT*(Q31@DIN)         ;MOD 2 ADDITION, IF /INIT
QC0  := QB+INIT                  ;SHIFT QB OR INITIALIZE
QC1  := (QC0+INIT)@QB            ;SHIFT QC0@QB OR INITIALIZE
QC2  := (QC1+INIT)@QB            ;SHIFT QC1@QB OR INITIALIZE
QC3  := QC2+INIT                 ;SHIFT QC2 OR INITIALIZE
QC4  := (QC3+INIT)@QB            ;SHIFT QC3@QB OR INITIALIZE
QC5  := (QC4+INIT)@QB            ;SHIFT QC4@QB OR INITIALIZE
QC6  := QC5+INIT                 ;SHIFT QC5 OR INITIALIZE
QC7  := (QC6+INIT)@QB            ;SHIFT QC6@QB OR INITIALIZE
QC8  := (QC7+INIT)@QB            ;SHIFT QC7@QB OR INITIALIZE
QC9  := QC8+INIT                 ;SHIFT QC8 OR INITIALIZE
QC10 := (QC9+INIT)@QB            ;SHIFT QC9@QB OR INITIALIZE
QC11 := (QC10+INIT)@QB           ;SHIFT QC10@QB OR INITIALIZE
QC12 := (QC11+INIT)@QB           ;SHIFT QC11@QB OR INITIALIZE
QC13 := QC12+INIT                ;SHIFT QC12 OR INITIALIZE
QC14 := QC13+INIT                ;SHIFT QC13 OR INITIALIZE
QC15 := QC14+INIT                ;SHIFT QC14 OR INITIALIZE
QC16 := (QC15+INIT)@QB           ;SHIFT QC15@QB OR INITIALIZE
QC17 := QC16+INIT                ;SHIFT QC16 OR INITIALIZE
QC18 := QC17+INIT                ;SHIFT QC17 OR INITIALIZE
QC19 := QC18+INIT                ;SHIFT QC18 OR INITIALIZE
QC20 := QC19+INIT                ;SHIFT QC19 OR INITIALIZE
QC21 := QC20+INIT                ;SHIFT QC20 OR INITIALIZE
QC22 := (QC21+INIT)@QB           ;SHIFT QC21@QB OR INITIALIZE
QC23 := (QC22+INIT)@QB           ;SHIFT QC22@QB OR INITIALIZE
QC24 := QC23+INIT                ;SHIFT QC23 OR INITIALIZE
QC25 := QC24+INIT                ;SHIFT QC24 OR INITIALIZE
QC26 := (QC25+INIT)@QB           ;SHIFT QC25@QB OR INITIALIZE
QC27 := QC26+INIT                ;SHIFT QC26 OR INITIALIZE
QC28 := QC27+INIT                ;SHIFT QC27 OR INITIALIZE
QC29 := QC28+INIT                ;SHIFT QC28 OR INITIALIZE
QC30 := QC29+INIT                ;SHIFT QC29 OR INITIALIZE
QC31 := QC30+INIT                ;SHIFT QC30 OR INITIALIZE

IF /OE2, SERDATAOUT := ENCDATAIN

ERRFLG := /Q0+/Q1+Q2+/Q3+/Q4+/Q5+/Q6+Q7+/Q8+Q9+/Q10+/Q11+/Q12+Q13
          +/Q14+/Q15+Q16+Q17+/Q18+Q19+Q20+Q21+A22+Q23+/Q24+/Q25
          +/Q26+Q27+Q28+Q29+/Q30+Q31
```

Table 3.

Configuration

The LCA device is manufactured with a programmable RAM-based CMOS process and must be configured upon power-up. This requires the loading of a configuration program into the LCA device. A number of methods for doing this exist. The Master Mode LOW was selected in this application.

In this mode, the LCA device is configured from an external EPROM containing the configuration data as shown in Figure 14. After an initial power-up delay and with the RESET pin HIGH, the LCA device reads the byte-wide configuration data from the EPROM starting at address 0000 and loads it into pins D0-D7. When the DONE pin goes HIGH, configuration is complete and the LCA device is free to perform as designed. The configuration process, lasts approximately 25 ms.

Summary

In this applications note, the principles of designing a Dual 32-bit Serial CRC in a Monolithic Memories' LCA device was described. The device implementation has many advantages which include its user-configurable logic functions, interconnects and I/O pins. This flexibility facilitates the design process significantly as standards in the data communications industry continually change. Thus, the LCA device is a viable solution for designers who wish to gain more control over their designs.

The Dual 32-bit CRC design developed for the LCA device is available upon request. The bit pattern and .LCA file will be provided for programming the LCA device in an EPROM. Please ask for design XDES09.LCA.

References

1. Nadia Sachs, *"Cyclic Redundancy Check using PAL® Devices."* AN-105, Monolithic Memories Inc., 2175 Mission College Blvd., Santa Clara, CA 95054-1592
2. Vivian Kong, *"Implementation of Serial/Parallel CRC Using PAL Devices."* AN-125, Monolithic Memories Inc., 2175 Mission College Blvd., Santa Clara, CA 95054-1592
3. IEEE Std 802.3-1985 *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,* The Institute of Electrical and Electronics Engineers, Inc., Wiley-Interscience, 1985
4. Robert Swanson, *"Understanding Cyclic Redundancy Codes,"* Computer Design, Nov. 1975.

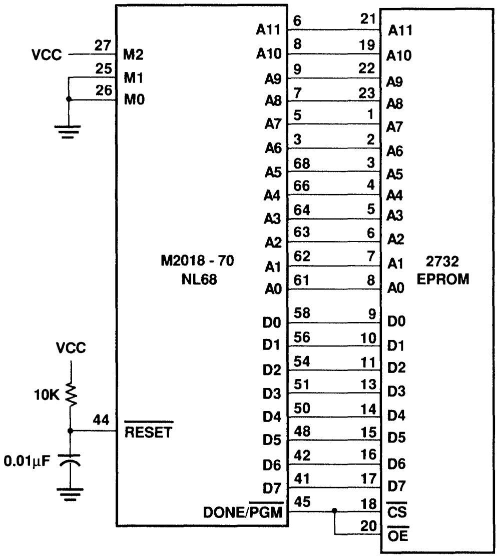


Figure 14. Master Mode Low Configuration for the Dual 32-Bit Serial CRC

Advanced
Micro
Devices
AN-178

LCA Device Implements an 8-Bit Format Converter in a PBX Switching Module

Raj Paripatyadar and C.B. Lee

Abstract

In Private Branch Exchanges, thousands of voice channels are multiplexed using time division multiplexing (TDM) techniques. Many of these voice channels may be multiplexed onto either serial or parallel channels. This application note describes an eight-bit format converter (parallel-to-serial and serial-to-parallel) circuit which transposes eight serial TDM highways into a single eight-bit parallel bus. A user-programmable Monolithic Memories' Logic Cell™ Array (LCA device) implements this circuit very efficiently using ninety-nine Configurable Logic Blocks (CLBs) out of one hundred CLBs available. Each CLB has a combinational logic section and a storage element. The circuit operates at up to 18.5 MHz when implemented in a 70-MHz CMOS, LCA M2018 device

LCA Device Implements an 8-Bit Format Converter in a PBX Switching Module

AN-178

Introduction

In a Private Branch Exchange (PBX) or Central Office (CO), incoming digitized voice circuits are "switched" or routed to their appropriate destination. The "switching" takes place while the data is in serial or parallel format. An eight-bit format converter circuit, widely used in a variety of PBX and CO architectures, translates the serial data streams into parallel data streams and vice-versa.

The eight-bit format converter circuit requires a large number of register elements. These elements, namely serial shift registers, can be implemented efficiently in an LCA device due to its high register content and flexible structure. The LCA M2018 programmable device contains one hundred configurable register elements, ninety-nine of which are used in the implementation of the converter circuit. This circuit is capable of running at 12.5 MHz and 18.5 MHz in a 50-MHz and 70-MHz LCA device, respectively.

Powerful software tools and circuit uniformity allowed the design to be laid out and simulated in just two days. However, if different support logic becomes necessary, PBX vendors are able to generate a new configuration program in the LCA reprogrammable device. Since these devices are manufactured in a CMOS technology, a complete PBX system with several LCA devices can help keep the active power consumption down. (Please refer to the LCA Design and Applications Handbook-Reference 3 for basic information regarding structure and programming aspects.)

Overview of a PBX

Central Offices (CO) and Private Branch Exchanges (PBX) provide both telephone and data services. A PBX may be considered as a localized telephone exchange serving the interconnection requirements of users in office environments. Outgoing calls are directed out of the PBX and routed through the local CO to the far destination. In general terms, a PBX or a CO consists of major blocks shown in Figure 1. Racks of peripheral equipment modules containing line cards provide access to telephone units, or to data adapter modules which provide data communication services. Trunk cards provide high-speed links to host computers, or other PBXs. Network switching modules interconnect calling parties to answering parties via digital multiplexed links. Control modules containing a Central Processor Unit (CPU), mass storage and local memory, perform the "setup" and "tear-down" of each circuit connection, as well as the monitoring of PBX activities. Central resources are modules of hardware and software which operate in a time-shared manner. The central resources include facilities such as ringing tone generator and message recording modules.

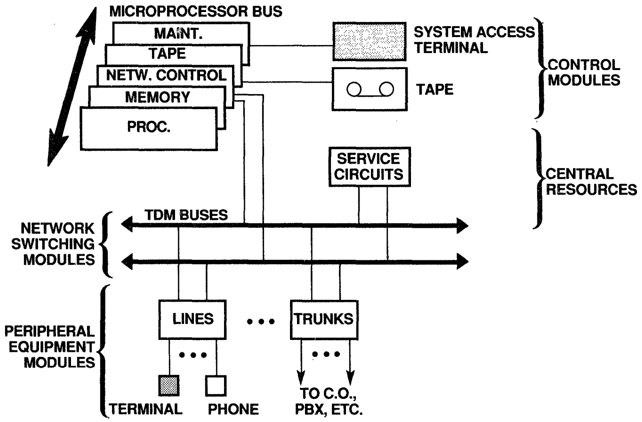


Figure 1. PBX Communication and Control Architecture

Hierarchy of Channel Multiplexing in a PBX

The main function of switching modules is to interconnect circuits between two or more parties such as in a conference call. A digital telephone unit, or a line card, contains a Coder/Decoder (CODEC) device. The CODEC device digitizes voice and transmits it into a Pulse Code Modulated (PCM) data stream on a common serial data highway at appropriate time intervals, which are assigned at the start of the call.

Analog voice is sampled at 8 KHz and passes through an eight-bit analog-to-digital converter. The digitized sample (eight-bit binary word) is transmitted serially to produce a serial data stream of (8 x 8 =) 64 Kbps. Thirty-two of these samples or CODECs are normally connected to a common highway operating at (32 x 64 =) 2.048 Mbps. Each CODEC is assigned "Serial Time Slots" of eight-bit duration, and sends its eight bits of data at the rate of 2.048 Mbps every 8 KHz. Another time slot is available on a separate highway for the receive data stream.

Therefore, a 2.048 Mbps highway can support thirty-two channels or voice connections in one direction. However, there are several ways to combine serial highways for additional voice connections. One way is to combine four highways into one high-speed highway of 8.192 Mbps. Further integration is possible, but the aggregate data rate becomes high and leads to implementation problems including timing, signal reflection, and radiation, to name a few.

Another way to integrate channels without increasing the clock rate is to convert eight serial data highways into a single eight-bit parallel bus. With this method, the serial clock rate and the parallel clock rate are the same, 2.048 Mbps. This means that data throughput on the eight serial highways of 16.384 Mbps is maintained in the eight-bit parallel bus operating at 2.048 Mbps. This works by careful alignment of the incoming parallel time slots to serial time slots on the serial highways. An eight-bit format converter scheme, sometimes called a "Corner Bender", is widely used in PBX or CO switching modules. Four of these parallel buses are combined (see Figure 2) to form a single parallel system bus that is thirty-two bits wide, yet still operates at 2.048 Mbps. The multiplexing architecture shown in Figure 2 is used in the Harris 20-20™ Integrated Network Switch (see Reference 1).

Space switching and/or time slot switching to interconnect different parties is performed either in the serial highway format or in the parallel data stream format.

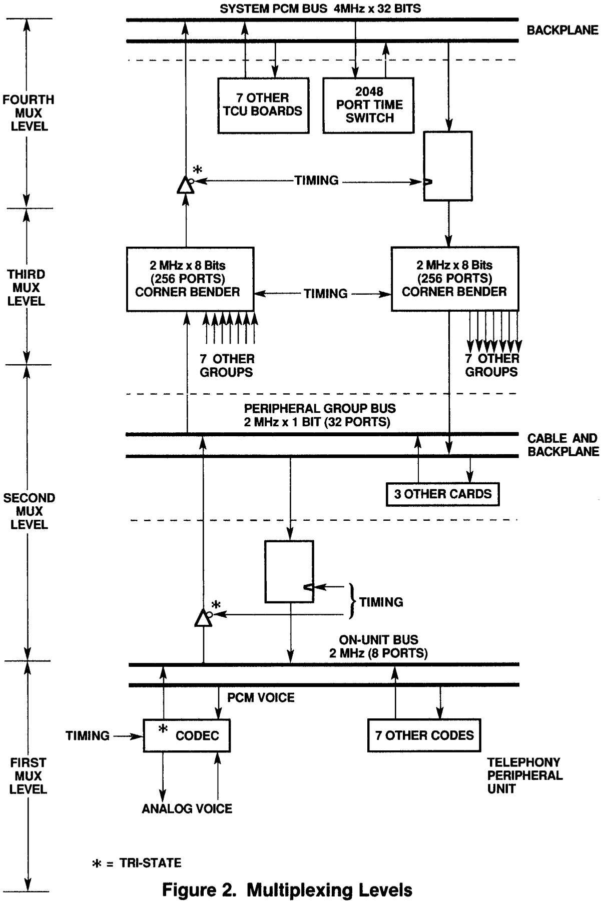


Figure 2. Multiplexing Levels

Description of the "Corner Bender" Circuit

The circuit consists of eight parallel-to-serial shift registers, and additional shift registers which provide proportional delays to synchronize the outputs. The principle of operation is shown in Figure 3, and the circuit is shown in Figure 4. Parallel data from the parallel bus is loaded with each clock into one of the eight shift registers in a rotating pattern. The data then shifts out of the shift register with each subsequent clock. After all eight registers have been loaded, the first register will be empty on the next clock. New data is loaded into this register at the same clock edge as the last bit shifts out to the next stage. Thus, a continuous flow of data through the registers is insured. The data out of the eight shift registers is skewed in relation to each other as shown in Figure 3b. The various space slots on the highways need to be aligned to keep synchronization. This is achieved by proportionally delaying the outputs. Channel 1 output is delayed by seven clocks (using a seven-bit serial shift register), channel 2 is delayed by six clocks, etc. Channel 8 output does not go through any additional delay. The final outputs of the delay registers are aligned and data is clocked out in a synchronous fashion. See Figure 3c.

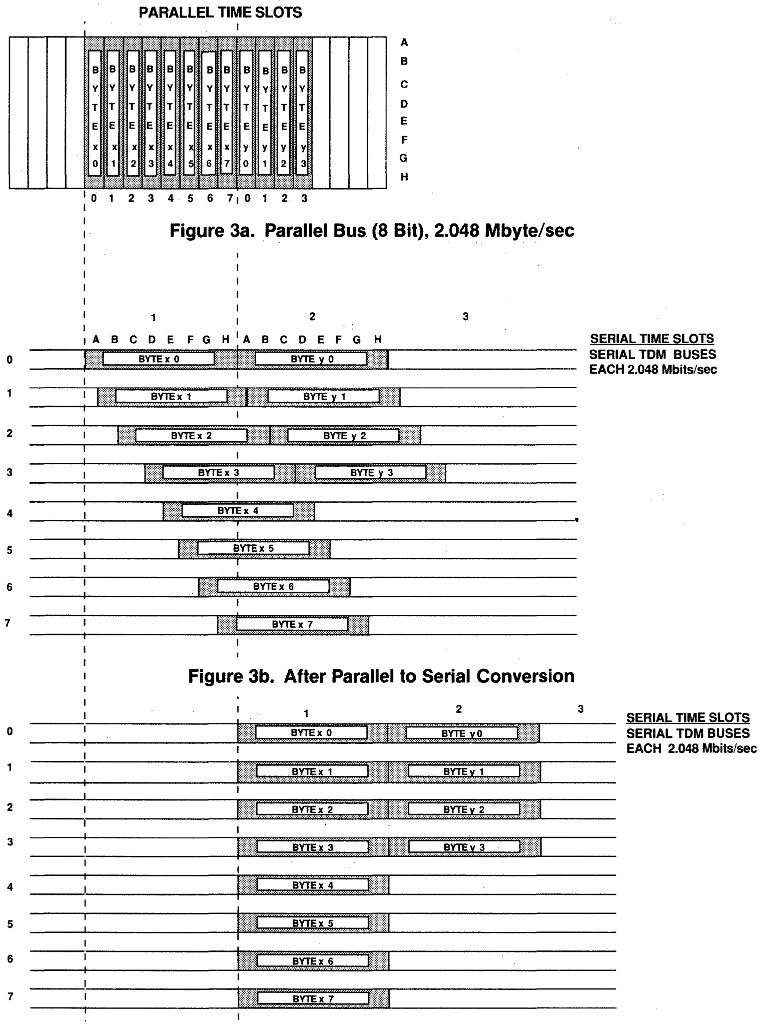


Figure 3a. Parallel Bus (8 Bit), 2.048 Mbyte/sec

Figure 3b. After Parallel to Serial Conversion

Figure 3c. After Appropriate Time Delay to Synchronize

Figure 3. Principles of "Corner Bending"

Figure 4 also shows the additional circuitry needed to control the shift registers. A preloadable three-bit counter keeps count of eight clock pulses. The parallel-to-serial bus conversion can be "programmed" to start in any register by setting the appropriate binary value on the counter preload inputs and applying a pulse to the sync input. If the loading sequence produced by the counter is not required, it can be disabled by connecting the "clock" to "sync" input. At each positive clock edge, the register loaded will depend upon the data at the counter inputs on the previous positive clock edge. The 3-to-8 decode circuit produces load pulses to latch parallel data into the shift registers. Figure 5 shows the parallel-to-serial conversion data matrix.

The description above shows the conversion of parallel data into eight streams of serial data. However, the same circuit also performs serial-to-parallel conversion. A serial eight-bit data stream on one of the eight inputs appears as an eight-bit parallel word on the eight outputs. Successive parallel words appearing at the eight outputs correspond to the serial data on each of the eight inputs in rotation. See Figure 5.

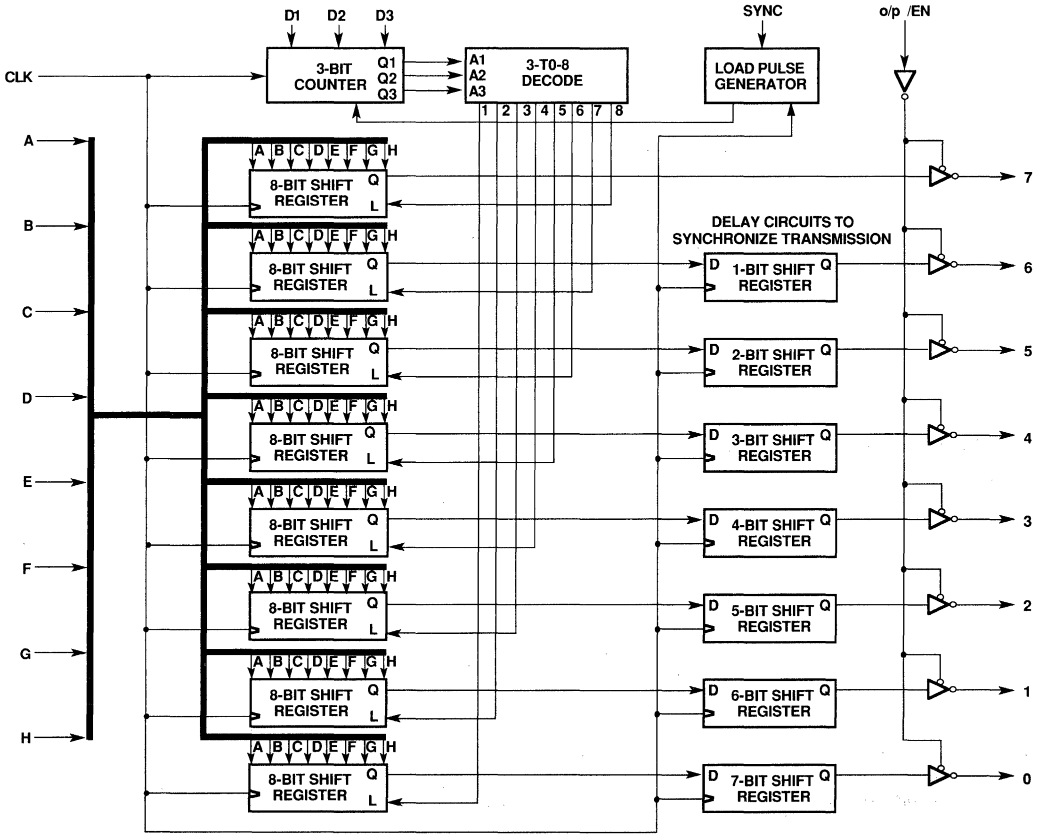
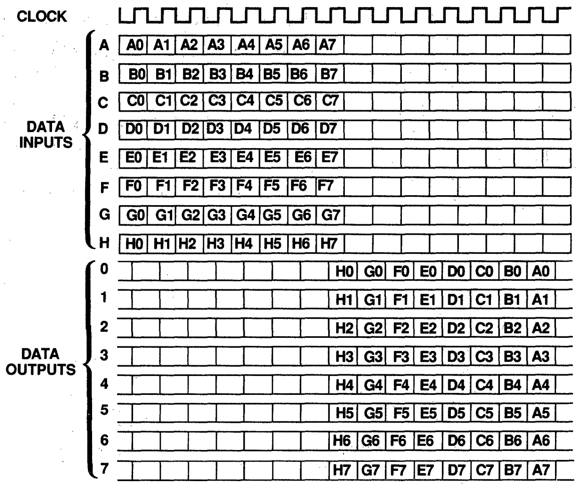


Figure 4. 8-Bit Format Converter ("Corner Bender") Circuit


CLOCK
DATA INPUTS
A A0 A1 A2 A3 A4 A5 A6 A7
B B0 B1 B2 B3 B4 B5 B6 B7
C C0 C1 C2 C3 C4 C5 C6 C7
D D0 D1 D2 D3 D4 D5 D6 D7
E E0 E1 E2 E3 E4 E5 E6 E7
F F0 F1 F2 F3 F4 F5 F6 F7
G G0 G1 G2 G3 G4 G5 G6 G7
H H0 H1 H2 H3 H4 H5 H6 H7
DATA OUTPUTS
0 H0 G0 F0 E0 D0 C0 B0 A0
1 H1 G1 F1 E1 D1 C1 B1 A1
2 H2 G2 F2 E2 D2 C2 B2 A2
3 H3 G3 F3 E3 D3 C3 B3 A3
4 H4 G4 F4 E4 D4 C4 B4 A4
5 H5 G5 F5 E5 D5 C5 B5 A5
6 H6 G6 F6 E6 D6 C6 B6 A6
7 H7 G7 F7 E7 D7 C7 B7 A7

PARALLEL TO SERIAL CONVERSION
A0, B0, C0, D0, E0, F0, G0, H0 = IN(I)0, IN(I)1, IN(I)2, IN(I)3, IN(I)4, IN(I)5, IN(I)6, IN(I)7
A1, B1, C1, D1, E1, F1, G1, H1 = IN(II)0, IN(II)1, IN(II)2, IN(II)3, IN(II)4, IN(II)5, IN(II)6, IN(II)7

SERIAL TO PARALLEL CONVERSION
A0, A1, A2, A3, A4, A5, A6, A7 = IN(I)0, IN(I)1, IN(I)2, IN(I)3, IN(I)4, IN(I)5, IN(I)6, IN(I)7
B0, B1, B2, B3, B4, B5, B6, B7 = IN(II)0, IN(II)1, IN(II)2, IN(II)3, IN(II)4, IN(II)5, IN(II)6, IN(II)7

*** IN(I)X ARE THE INPUT 8-BIT WORDS**

Figure 5. Data Conversion

Comparison of Implementation Techniques

The "Corner Bender" circuit can be implemented in several ways. The efficiency of an LCA design can be compared to that of a design in SSI/MSI logic devices or Erasable Programmable Logic devices (EPLDs). Table 1 contains the comparison of package counts with the different alternatives.

The circuit implemented in 74LSxx devices requires nineteen packages. Alternatively, sixteen simple PLD devices (such as a PAL20R8 or a PAL20R4) are required to implement the same circuit. Since the circuit is register intensive and not designed for high speed, PLDs are not chosen for this application.

The functionally larger EPLD devices, EP1200 and EP1800, are more register intensive and contain twenty-eight and fourty-eight register elements, respectively. Nevertheless, four EP1200 devices, or two EP1800 plus a smaller PLD device, is required to implement this design.

The entire Corner Bender circuit fits neatly into a single LCA 2018 device. It uses ninety-nine out of the one hundred available internal macrocells and demonstrates efficient implementation of shift registers and small counters. An LSI device from Plessey (Reference 2) is available to implement the same circuit. However, it is an NMOS device and limited to speeds of under 2 MHz.

EQUIVALENT CIRCUIT IN SSI/MSI DEVICES		PACKAGES
Parallel in serial out shift Registers	74LS165	8
Serial in Parallel out shift Registers	74LS164	7
4-bit counter	74LS161	1
3 to 8 decoder	74LS138	1
Three-state buffers	74LS241	1
EQUIVALENT CIRCUIT IN EPLDs		
a) EP1200	28 macro cells each	4
b) EP1800	48 macro cells each	2
plus PAL device e.g., 16R4		1
EQUIVALENT CIRCUIT IN MMI LCA DEVICE		
LCA 2018 device	100 macro cell	1

Table 1. Implementation Alternatives for the 8-Bit Format Converter Circuit "Corner Bender"

"Corner Bender" Design in an LCA Device

A count of the register elements in the circuit shows a need of sixty-four elements for the eight-shift registers, twenty-eight elements for the delay registers, three elements for the counter, and four elements for the 3-to-8 decode circuit. Each configurable logic block (CLB) in the LCA device can produce two outputs. Hence, only four CLBs are required for the 3-to-8 decode circuit. Since, each CLB contains one register element, the total count of CLBs is therefore ninety-nine. The LCA 2018 contains one hundred CLBs. Twenty-two input/output blocks (IOBs) were used. Figure 6 shows the layout of the design in the LCA device. The circuit fits into a 68-pin PLCC package.

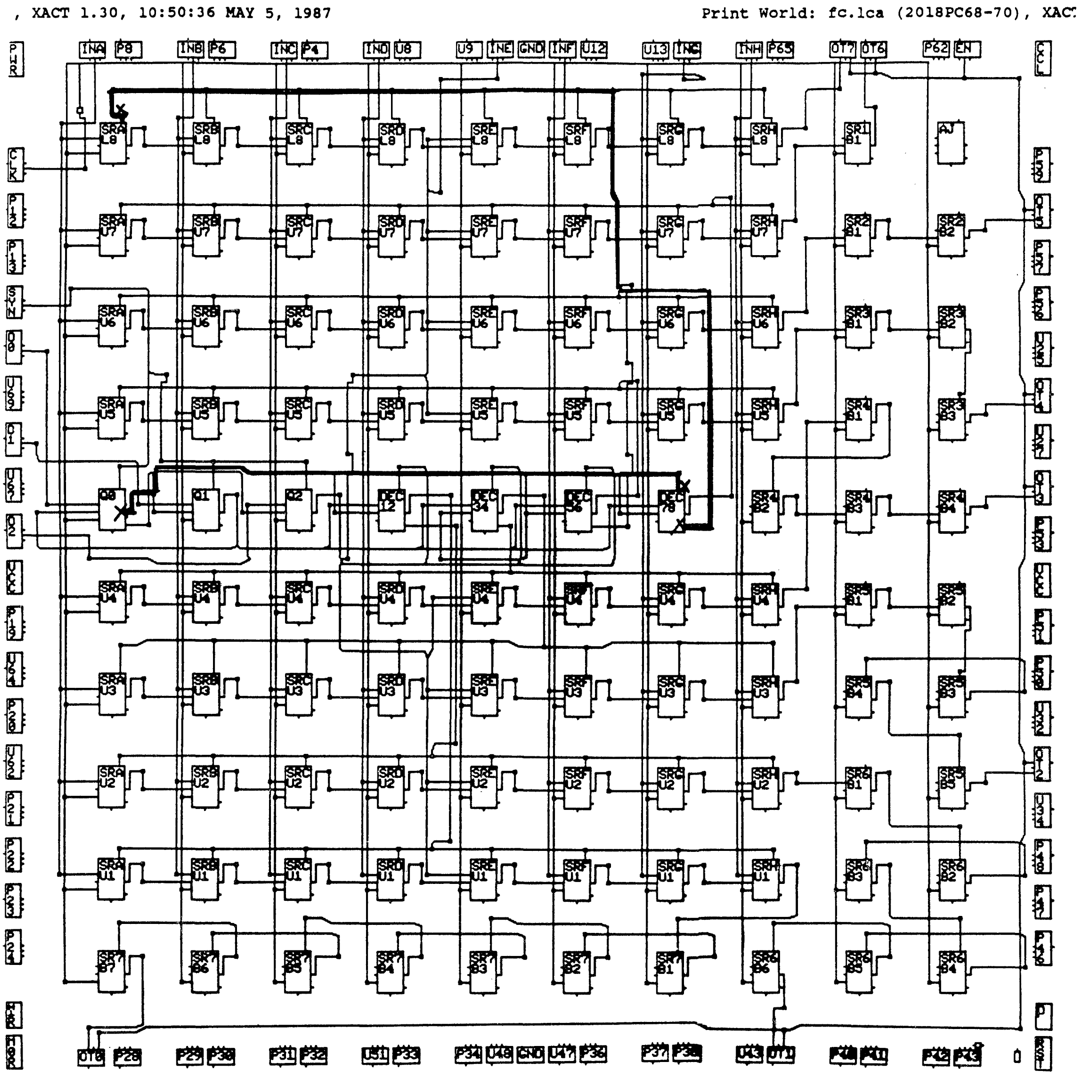


Figure 6. Layout Diagram of "Corner Bender" Circuit on LCA Device of Size 10x10

The Monolithic Memories' XACT™ Design Editor is used to create the design by implementing the appropriate logic functions in CLBs and IOBs. An example of a CLB for one bit of the eight-bit shift register in this design is shown in Figure 7. Input A is the decoder input and it is ANDed with input B, the data to be shifted. To implement the eight-bit shift registers from the one-bit shift register, eight of the one-bit shift registers were linked together so that the output of each register became an input to the next register. This is shown in Figure 8 where each CLB represents one bit of the eight-bit shift register.

The three-bit counter and the 3-to-8 decoder in this design were implemented in CLBs as shown in Figures 9 and 10. The counter is a synchronous binary counter with ripple carry and parallel load. The decoder is a standard 3-to-8 decoder. The outputs of the counter become inputs to the decoder, whereas the outputs of the decoder are used to decode the eight 8-bit shift registers.

With the XACT Development System, the designer can optimally arrange the logic blocks on the LCA device in order to minimize net delays between each block. With this in mind, the layout for the design is described below:

Four of the eight-bit parallel-to-serial shift registers were placed starting from the top left-hand edge of the LCA device (see Figure 7). The three-bit counter (three CLBs) and the 3-to-8 decoder (four CLBs) were then placed on the following row. The next four rows contained the last four parallel-to-serial shift registers. This allowed the shift register select lines to have minimal delay spread when accessing all eight shift registers. The seven serial-to-serial shift registers were placed in the remaining CLBs as uniformly as possible.

The optimum placement and distribution of configured blocks in the array is influenced by the performance needs of the system. Blocks placed in close proximity can use local interconnection resources which incur short signal propagation delays, whereas blocks placed further apart must use either "long lines" or other interconnection resources. Manual optimization using the delay efficient "long lines" was performed for the most critical net connections. After routing completion, the longest delay between two clock pulses was the delay for the counter to change state, the state which is decoded via the 3-to-8 decoder and selects the appropriate shift register to load the parallel data (see Figure 7). This delay was 54 ns, 79 ns, and 106 ns, respectively for the 70-, 50-, and 33-MHz versions of the device. The delays were measured using the XACT simulation package by invoking the timing delay calculator. This translates to a maximum circuit operating speed of about 18.5 MHz for the 70-MHz version of the device, or 9.43 MHz for the 33 MHz version.

Although fabricated in a CMOS technology, the inputs to the LCA device can be made either TTL or CMOS compatible. For high fan-out CMOS or LS TTL-compatible loads, the output buffers of the LCA device are capable of driving 4 mA. Moreover, each output buffer can be put into a HIGH-Z state for bus-driving applications. This features was also used in the design of the eight-bit format converter.

More information about entering a design with the XACT Development System is included in the LCA Design and Applications Handbook (Reference 3). Information about configuring the LCA device is described in the "Configuring the LCA Device" applications note.

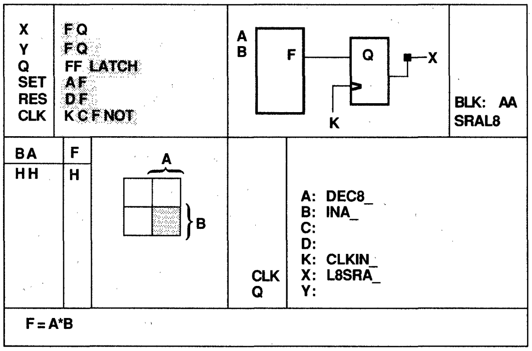


Figure 7. One Bit of an 8-Bit Serial - Parallel Shift Register

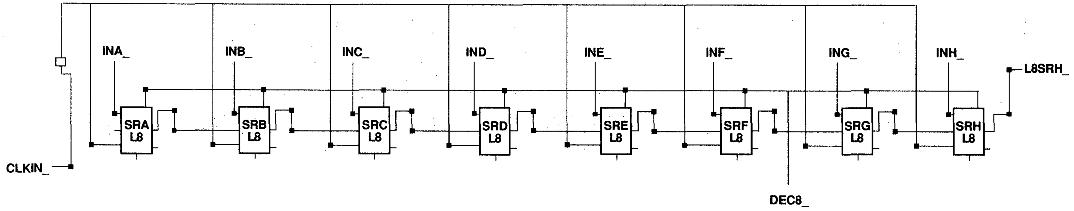


Figure 8. CLB Schematic Output for the 8-Bit Shift Register

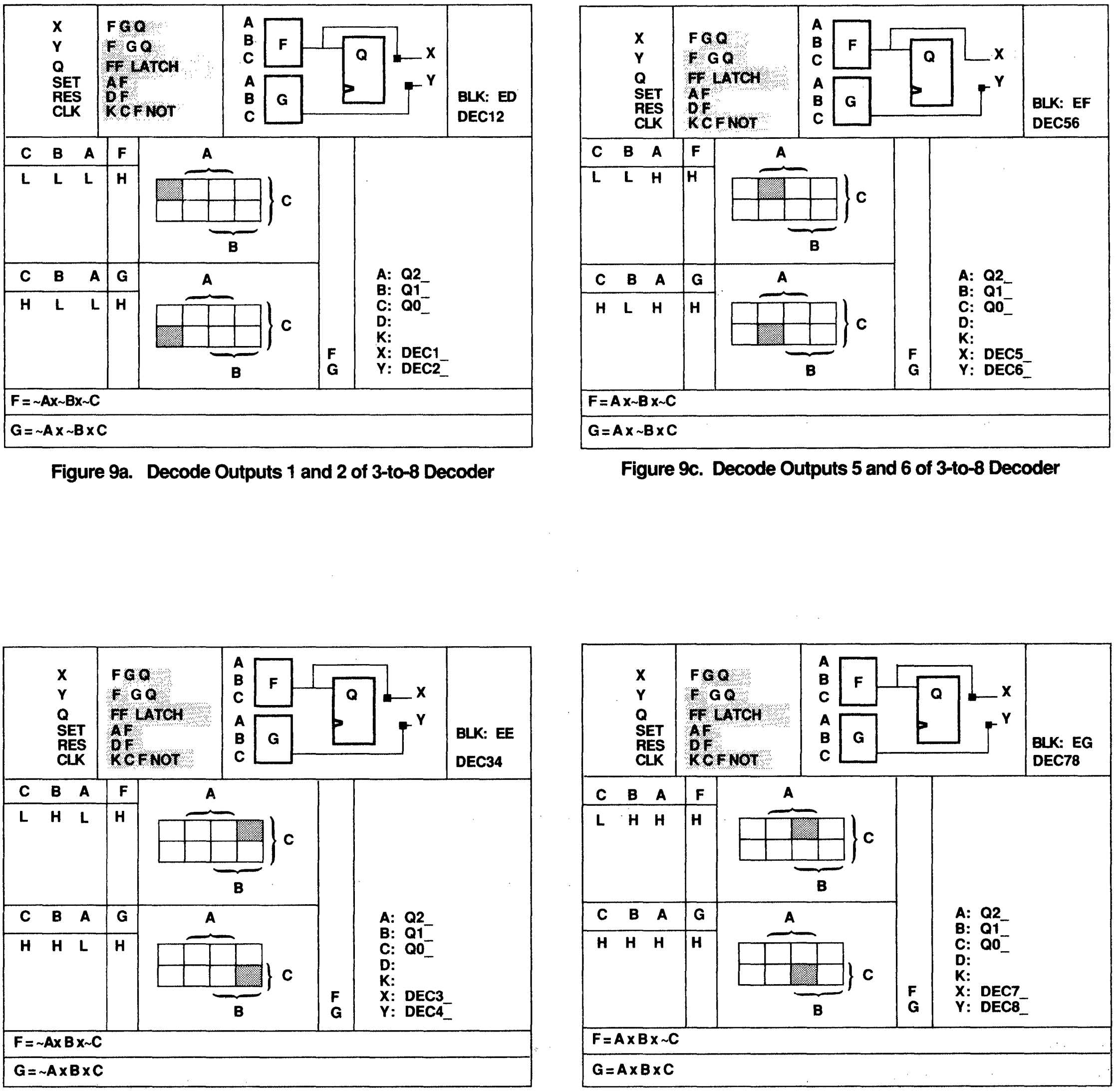


Figure 9a. Decode Outputs 1 and 2 of 3-to-8 Decoder

Figure 9c. Decode Outputs 5 and 6 of 3-to-8 Decoder

Figure 9b. Decode Outputs 3 and 4 of 3-to-6 Decoder

Figure 9d. Decode Outputs 7 and 8 of 3-to-8 Decoder

Figure 9. CLB Configuration of a 3-to-8 Decoder

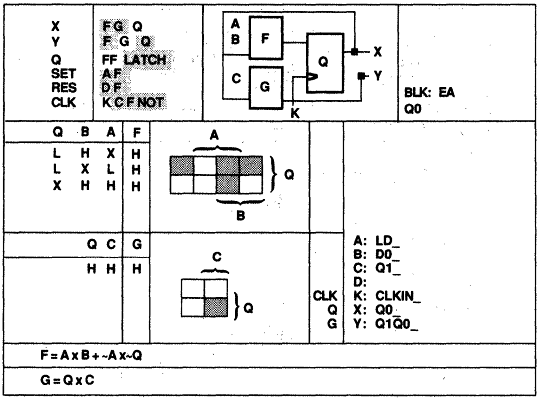


Figure 10a. 1st Bit of a 3-Bit Counter

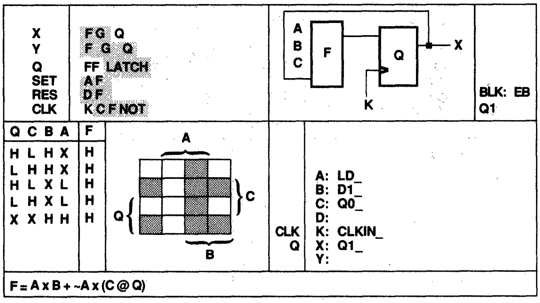


Figure 10b. 2nd Bit of a 3-Bit Counter

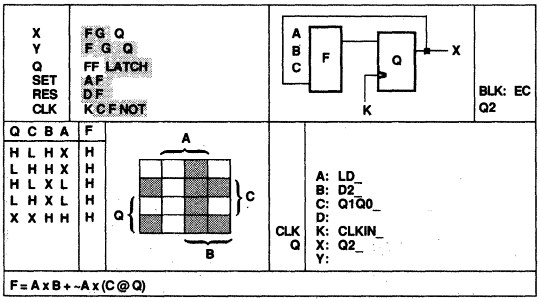


Figure 10c. 3rd Bit of a 3-Bit Counter

Figure 10. CLB Configuration of a 3-Bit Binary Counter with Ripple Carry

Summary

The LCA device has achieved a good solution to the principles used for multiplexing digitally-encoded voice channels in both serial and parallel hierarchies. These hierarchies can accommodate thousands of voice circuits in a PBX switching module. The detailed design of the eight-bit (parallel-to-serial and serial-to-parallel) format converter circuit, or Corner Bender, developed in Monolithic Memories' LCA M2018 device was shown to be efficiently implemented. Despite the fast development of the system using the XACT Design Editor, the final design was programmed and bench tested to verify functional integrity.

The Corner Bender design is available from Monolithic Memories upon request. The bit pattern and .LCA file will be provided for programming the LCA device in an EPROM. Please ask for design XDES05.LCA

References

1) *"Harris 20-20 Integrated Network Switch",* A. Jackson, IEEE SAC-3, No. 4 July 1985, p561-568.

2) MJ1410, 8-bit Format Converter, Data Sheet, Plessey Semiconductors.

3) Logic Cell Array Design and Applications Handbook, Monolithic Memories, 1987.

Reconfigurable Programmable Devices (LCA) Simplify Digital TDM Line Transcoder

C. B. Lee and Cindy Lee

Abstract

The Logic Cell™ Array (LCA) is a high-density CMOS programmable integrated circuit. Its reprogrammability and reconfigurability make complex design of custom LSI devices easy and flexible.

This application note describes the design of a universal transcoder on trunk transmission lines. Any one of three different transcoders (B8ZS, B6ZS, and HDB3) can be implemented in a single LCA device (M2018). Because it is reconfigurable, any one of these different transcoders may be selected. Other common communication features can be implemented simply by reconfiguring the Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs), or interconnect.

Reconfigurable Programmable Devices (LCA) Simplify Digital TDM Line Transcoder

C. B. Lee and Cindy Lee

TDM Hierarchies

Recently, the International Telegraph and Telephone Consultative Committee (CCITT) agreed on transmission hierarchies based on different national standards. Two major hierarchies are in use today. The first hierarchy, known as the T-carrier, is used in North America and Japan and multiplexes twenty-four voice channels together using Time Division Multiplexing (TDM) technique. T-carrier is based on a primary rate of 1.544 Mbps and multiples up to 274.176 Mbps (see Figure1).

The other hierarchy, used in Europe, Africa, and South America, multiplexes thirty voice channels and two signaling channels together. This TDM hierarchy is based on a primary rate of 2.048 Mbps and multiples up to 139.264 Mbps (see Figure 2) .

CCITT recommendation G.703 defines line coding for both TDM hierarchies with different orders. Table 1 shows a 1.544 Mbps primary rate T-carrier hierarchy with four different orders. Alternate Mark Inversion (AMI) or Binary 8 Zeros Substitution (B8ZS) is the proper line coding in the first order. In the second order, if the transmission medium is coaxial pair cables, then B8ZS is used. Otherwise, Binary 6 Zeros Substitution (B6ZS) is used for symmetrical pair cables. In the third order, Binary 3 Zeros Substitution (B3ZS) is used, while polar binary NRZ is used in the fourth order.

Similarly, Table 2 shows line coding based on a 2.048 Mbps primary rate with four different orders. High Density Bipolar (HDB3) is used in the first, second, and third orders. Finally, Code Mark Inversion (CMI) is used in the fourth order.

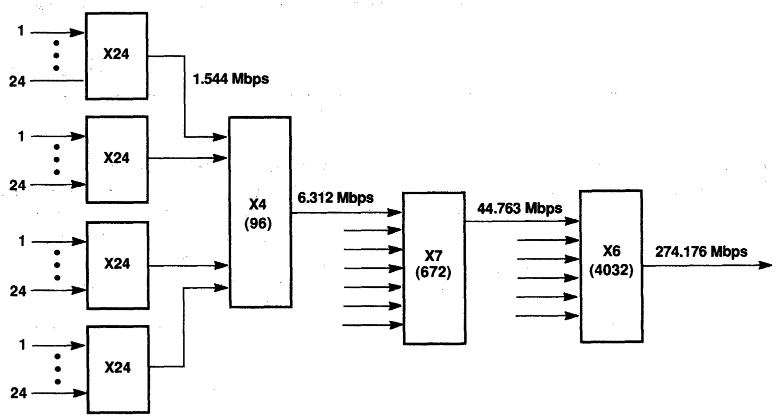


Figure 1. TDM Hierarchy at 1.544 Mbps Primary Rate

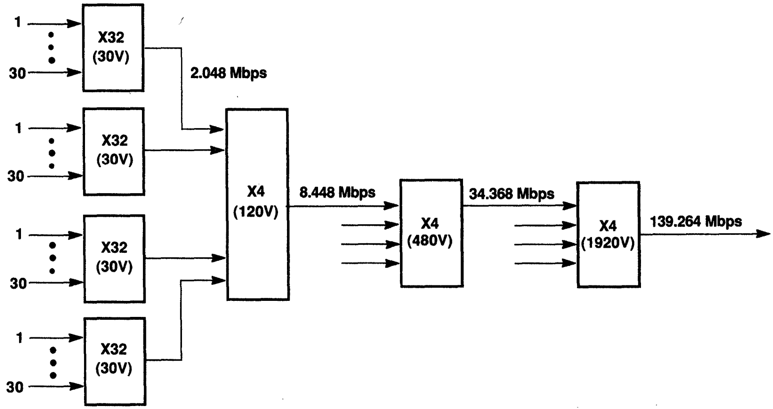


Figure 2. TDM Hierarchy at 2.048 Mbps Primary Rate

ORDER	CHANNEL BANK	NUMBER OF VOICE CHANNELS	BIT RATE (Mbps)	LINE CODING
1	DS-1	24	1.544	AMI or B8ZS
2	DS-2	96	6.312	B8ZS-Coaxial Pair B6ZS-Symmetric Pair
3	DS-3	672	44.736	B3ZS
4	DS-4	4032	274.176	Polar Binary

Table 1. Line Coding for TDM Hierarchy at 1.544 Mbps Primary Rate

ORDER	NUMBER OF VOICE CHANNELS	BIT RATE (Mbps)	LINE CODING
1	30	2.048	HDB3
2	120	8.448	HDB3
3	480	34.368	HDB3
4	1920	139.264	CMI (2-level NRZ)

Table 2. Line Coding for TDM Hierarchy at 2.048 Mbps Primary Rate

Binary Codes

Unipolar binary codes form the information exchange in computer systems from one device to another over short distances. Unipolar binary codes work well at short distances but DC wander and a lack of timing information make them unsuitable for long transmission lines. DC wander is caused from unipolar binary signals being transmitted via a certain media over long-distance transmission lines. The signals will be attenuated after a few kilometers.

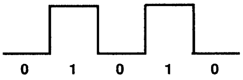


Figure 3a. Unipolar Binary Codes

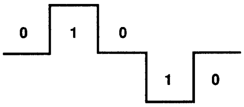


Figure 3b. Bipolar Codes

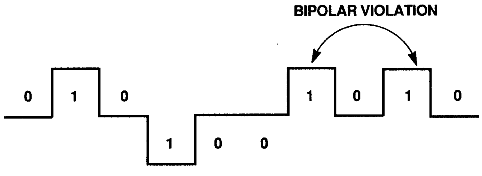


Figure 4. Bipolar Violation

Bipolar Codes

Unipolar binary codes require two voltages to represent a binary state (see Figure 3a). On the other hand, bipolar binary codes, also known as AMI codes, need three different voltages (+5 V, 0 V, and -5 V) to represent the same binary condition: logic zero or logic one (see Figure 3b). By making logic one signals alternate between +5 V and -5 V, DC wander is eliminated on a long transmission line because the mean DC level is integrated to zero volts. If, however, two continuous ones have the same polarity a bipolar violation occurs (see Figure 4). A long string of ones of bipolar codes can provide timing information but a long string of zeros cannot. To counteract this problem, several modified bipolar codes provide timing information for a long string of logic zeros.

Modified Bipolar Codes

The B8ZS, B6ZS, and HDB3 codes are all modified bipolar codes. They provide timing information for a long string of zeros by forcing bipolar violations.

B8ZS line coding encodes logic ones as alternating positive (+5 V) and negative (-5 V) signals. It substitutes any continuous sequence of eight zeros with a special pattern (see Table 3). Depending on the preceding polarity, one of two B8ZS substitution codes is generated: 000+-0-+ or 000-+0+-. The "+" indicates positive voltage, "-" indicates negative voltage, and "0" indicates zero voltage. If the preceding polarity is positive, 000+-0-+ is generated; otherwise, 000-+0+- is generated. The B8ZS substitution code forces bipolar violations in the fourth and the seventh bit positions.

B6ZS is similar to B8ZS. Rather than detecting eight continuous zeros, B6ZS substitutes for six continuous zeros with a special pattern. The B6ZS substitution codes are 0+-0-+ and 0-+0+-, depending on the preceding polarity. B6ZS substitution code forces the second and fifth zeros to bipolar violation (see Table 4).

PRECEDING POLARITY	B8ZS SUBSTITUTION
+	000+-0-+
-	000-+0+-

Table 3. B8ZS Substitution Table

PRECEDING POLARITY	B6ZS SUBSTITUTION
+	0+-0-+
-	0-+0+-

Table 4. B6ZS Substitution Table

The European HDB3 line coding also uses alternating positive and negative signals to represent logic ones. It also substitutes any four continuous zeros with a special HDB3 substitution code. This HDB3 substitution code forces the fourth bit to be a bipolar violation. The four possible HDB3 substitution codes are: +00+, 000+, -00-, and 000-. The choice of substitution code depends on the preceding polarity and the number of logic ones after the preceding bipolar violation. When the number of logic ones is odd, 000V is generated. Otherwise, B00V is generated. "V" indicates the bipolar violation while "B" indicates the bipolar signal. The polarity of V in 000V is the same as that of the preceding logic one signal. If the HDB3 code is B00V, the polarity of B is opposite that of the preceding logic one signal, and the polarity of V is the same as B (see Table 5).

PRECEDING POLARITY	NUMBER OF LOGIC ONES AFTER THE PREVIOUS BIPOLAR VIOLATION	
	ODD	EVEN
Positive +	000+	-00-
Negative -	000-	+00+

Table 5. HDB3 Substitution Table

Logic Cell Array Implementation

The Logic Cell Array (LCA) device is a high-density CMOS integrated circuit. Its user-programmable static RAM array architecture consists of Input/Output Blocks (IOBs), Configurable Logic Blocks (CLBs), and Interconnect. All of them are fully user-programmable and user-reconfigurable. High-density, programmability and reconfigurability of LCA devices allow customer-defined LSI functions to be incorporated and modified at low cost and with fast turnaround.

Currently, two LCA devices are available, their complexity is based on the number of CLBs within the device. All CLBs are arranged in a matrix in the center of the device. The M2064 has sixty-four CLBs which are arranged in an 8-row by 8-column matrix. The M2018 has one hundred CLBs arranged as a 10 by 10 matrix.

A CLB can be configured to be a storage element (registered or latched), a combinational logic block or a mixed combinational logic block with storage element. Each CLB has four general-purpose inputs (A, B, C, and D); and a special clock input (K). Also, each CLB can perform any function of up to four variables or any two functions of up to three variables. These variables can be selected from external inputs or from the internal register feedback path.

Initially, an attempt was made to design the line transcoder with the M2064 device. However, all three line transcoder designs required more than sixty-four CLBs each. Fortunately, it is easy to convert a design from M2064 to M2018 because the change is made just by selecting the M2018 part type in the convert command when using the XACT Design Editor. Any of the preceding line coding schemes can be implemented in a single M2018 because of its higher density (100 CLBs). Since the LCA device is reconfigurable, a line transcoder implemented in a single LCA device (M2018),can support the North American T1 carrier (B8ZS line coding) or T2 carrier (B6ZS line coding), or even the European standard (HDB3 line coding).

All three line transcoders can be configured and have the same pinout assignments. This offers a simple solution for supporting the different standards without component replacement. Once a design has been established, "on-the-fly" modification is possible through the reconfigurability of the component. For example, if the communication medium is changed in the T2

PIN NUMBER	DESCRIPTION	B8ZS	B6ZS	HDB3
Pin 01	GND	GND	GND	GND
Pin 11	Clock	CLK	CLK	CLK
Pin 12	Master Reset	MR	MR	MR
Pin 13	Positive input for decoder	IPB	IPB	IPH
Pin 14	Positive output from encoder	OPB	OPB	OPH
Pin 15	NRZ input	INR	INR	INR
Pin 17	Negative input for decoder	INB	INB	INH
Pin 18	V_{CC}	V_{CC}	V_{CC}	V_{CC}
Pin 19	Negative output from encoder	ONB	ONB	ONH
Pin 34	NRZ output	ONR	ONR	ONR
Pin 35	GND	GND	GND	GND
Pin 52	V_{CC}	V_{CC}	V_{CC}	V_{CC}
Pin 56	Error*	ERR	ERR	ERR
Pin 57	Bipolar violation error	BVP	BVP	BVP
Pin 59	Special pattern detected	B8Z	B6Z	HDB

* Means that the ERROR is asserted when both positive and negative signals are high simultaneously.

Table 6. Cross Reference of Pin Names for Three Line Transcoders

standard, the transcoder may be reprogrammed from B8ZS to B6ZS, or vice versa, without any other alterations to the hardware. In total, fifteen pins are used in these designs; four power pins, two central control pins, three pins for the encoder, and six pins for the decoder. Table 6 cross references the pin names for the three different line transcoders.

B8ZS Line Coding

A B8ZS encoder converts NRZ data to B8ZS code. The B8ZS encoder consists of a 3-CLB, 3-bit counter, a 5-CLB, 5-bit shift register, a 6-CLB sequence state machine, and a 13-CLB encoding state machine (see Figure 5). A 3-bit counter detects eight continuous zeros. A 5-bit shift register delays the NRZ input data to synchronize with the 3-bit counter's outputs. The sequence state machine provides the sequence state (Q2, Q1, and Q0) for encoding state machine. The encoding state machine generates the B8ZS code. It is a pair of signals which interface to line driver to provide three level signals: positive (PB8ZS), negative (NB8ZS), and zero signals.

A B8ZS decoder detects B8ZS code on a pair of positive and negative B8ZS signals (see Figure 6). When either of the following sequences: 000+-0-+ or 000-+0+- is detected, the decoder generates a string of eight zeros. Otherwise, the encoder converts bipolar signals to binary NRZ data.

Figure 7 is a block diagram of the B8ZS decoder. It includes a 19-CLB decoding state machine, an 8-CLB, 8-bit shift register, a 5-CLB RESET generator, and a 7-CLB bipolar violation and error flag generator. Four state variables (HQ3, HQ2, HQ1, and HQ0) are generated from the decoding state machine. The 8-bit shift register buffers DSOUT data in order to provide eight continuous zeros when the B8ZS signal is asserted. The B8ZS signal is generated when the B8ZS substitution code is detected. The bipolar violation error (BVP) occurs when any two sequential signals of the same polarity are detected except the B8ZS substitution code. If both positive B8ZS and negative B8ZS are detected simultaneously, the error signal (ERROR) is generated.

The implementation details of the B8ZS line transcoder within a single M2018 are shown in Figure-8. This design requires sixty-six out of the one hundred available CLBs.

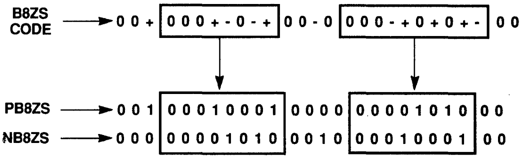


Figure 6. B8ZS Codes in Positive-B8ZS and Negative-B8ZS Signals

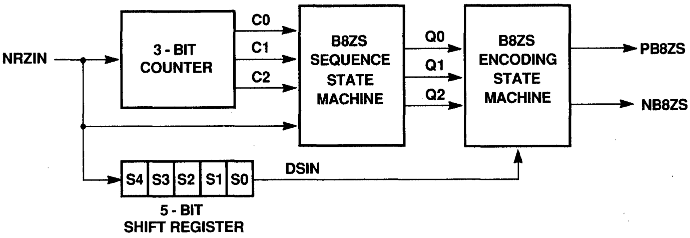


Figure 5. Block Diagram of the B8ZS Encoder

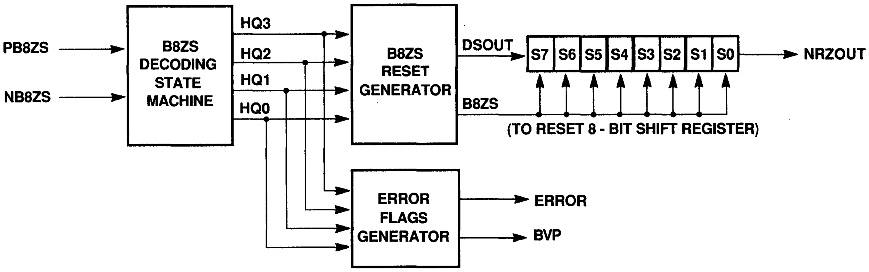


Figure 7. Block Diagram of the B8ZS Decoder

Print World: B8ZS.LCA (2018PC68-70), XACT 1.30, 10:10:45 MAY 12, 1987

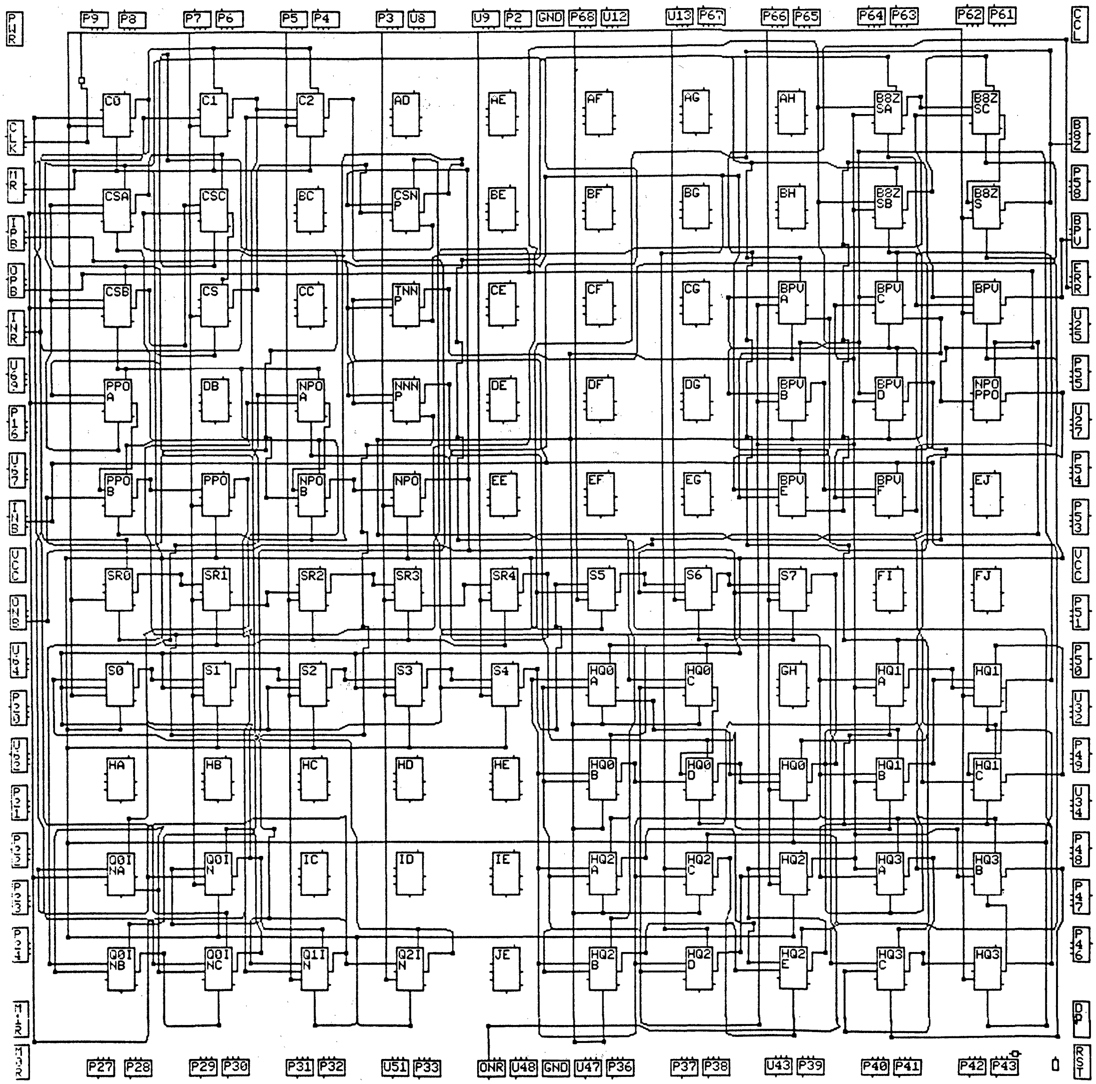


Figure 8. B8ZS Line Transcoder LCA Design

B6ZS Line Coding

The M2018 implementation of the B6ZS line transcoder is shown in Figure 9. In total, sixty-seven CLBs are used to create the design. The single difference between a B8ZS encoder and a B6ZS encoder is the count for continuous zeros. While the 3-bit counter counts eight continuous zeros for the B8ZS encoder, it counts six for the B6ZS encoder. The B8ZS encoder consists of a 5-CLB, six-zeros detector, a 5-CLB, 5-bit shift register, a 6-CLB sequence state machine, and a 13-CLB encoding state machine.

A B6ZS decoder detects two B6ZS substitution codes: 0+-0-+ or 0-+0+-. When one of those two B6ZS substitution codes is detected, the decoder generates six continuous zeros. Otherwise, the decoder converts bipolar signals to binary NRZ data. The B6ZS decoder includes a 20-CLB decoding state machine, a 6-CLB, 6-bit shift register, a 5-CLB RESET generator, and a 7-CLB bipolar violation and error flag generator. The 6-bit shift register is needed to buffer DSOUT data in order to provide six continuous zeros.

Print World: B6ZS.LCA (2018PC68-70), XACT 1.30, 10:25:09 MAY 12, 1987

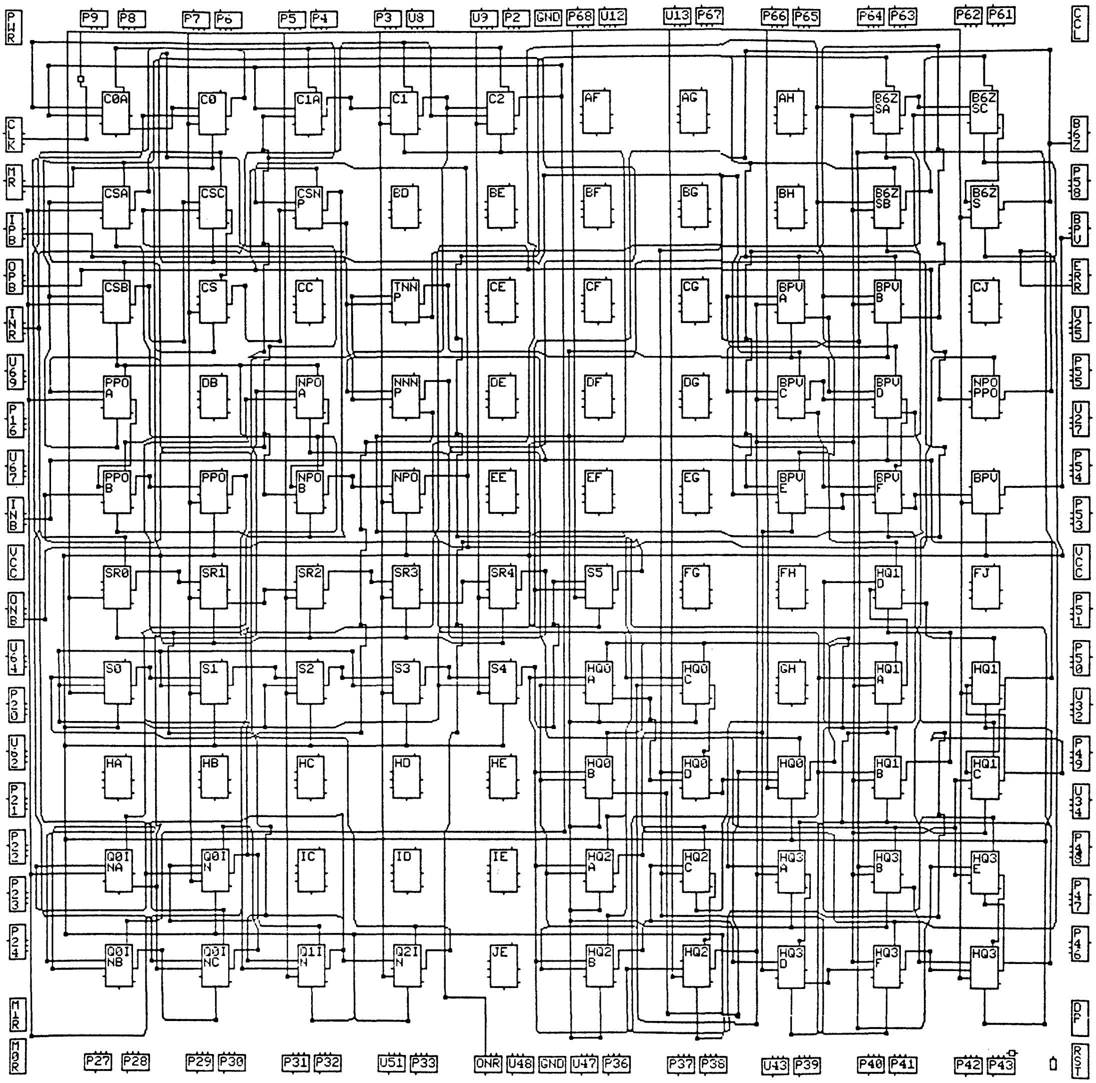


Figure 9. B6ZS Line Transcoder LCA Design

HDB3 Line Coding

In Figure 10, the HDB3 encoder includes a 4-CLB, 4-bit shift register, a 5-CLB HDB3 signal generator, and three-state machines. These state machines are, a 3-CLB ODD_EVEN state machine, a 5-CLB DETECT_4_ZEROS state machine, and 23-CLB encoding state machine. The ODD_EVEN state machine detects the polarity of the preceding pulse. The DETECT_4_ZEROS state machine detects four continuous zeros. The 4-bit shift register delays NRZIN data to synchronize with the output of DETECT_4_ZEROS state machine. The encoding state machine provides four encoding states variables (Q3, Q2, Q1, and Q0) to generate two HDB3 signals: the positive HDB3 signal (PHDB) and negative HDB3 signal (NHDB).

A HDB3 decoder detects four HDB3 substitution codes: +00+, 000+, -00-, and 000-. When any of these four HDB3 substitution codes is detected, the decoder generates four continuous zeros. Otherwise, the decoder converts bipolar signals to binary NRZ data. Figure 11 is a block diagram of the HDB3 decoder. It includes a 23-CLB decoding state machine, a 4-CLB, 4-bit shift register, a 3-CLB RESET generator, and a 2-CLB bipolar violation and error flags generator. Four state variables (HQ3, HQ2, HQ1, and HQ0) are generated from the decoding state machine. The 4-bit shift register buffers DSOUT data in order to provide four continuous zeros when the HDB3 signal is asserted. The HDB3 signal is generated when the HDB3 substitution code is detected. The bipolar violation error (BVP) occurs when any two sequential signals of the same polarity are detected except the HDB3 substitution code. If both positive HDB3 and negative HDB3 are detected simultaneously, the error signal (ERROR) is generated. The M2018's layout of the HDB3 line transcoder is shown in Figure 12. This design requires seventy-two out of the one hundred available CLBs.

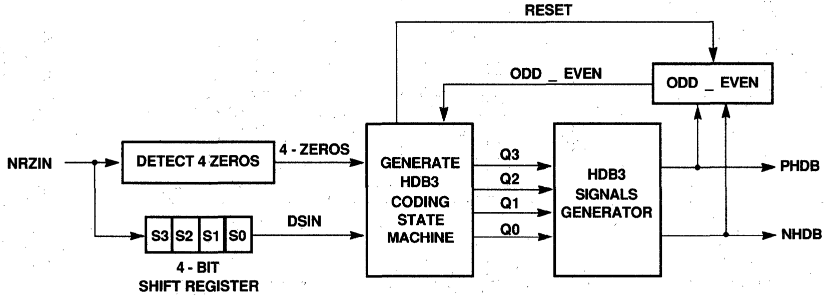


Figure 10. Block Diagram of the HDB3 Encoder

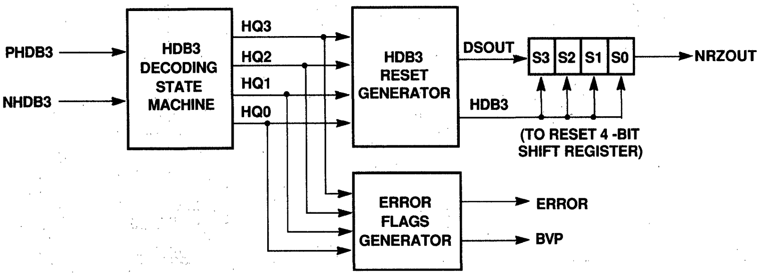


Figure 11. Block Diagram of the HDB3 Decoder

Print World: HDB3.LCA (2018PC68-70), XACT 1.30, 09:36:22 MAY 12, 1987

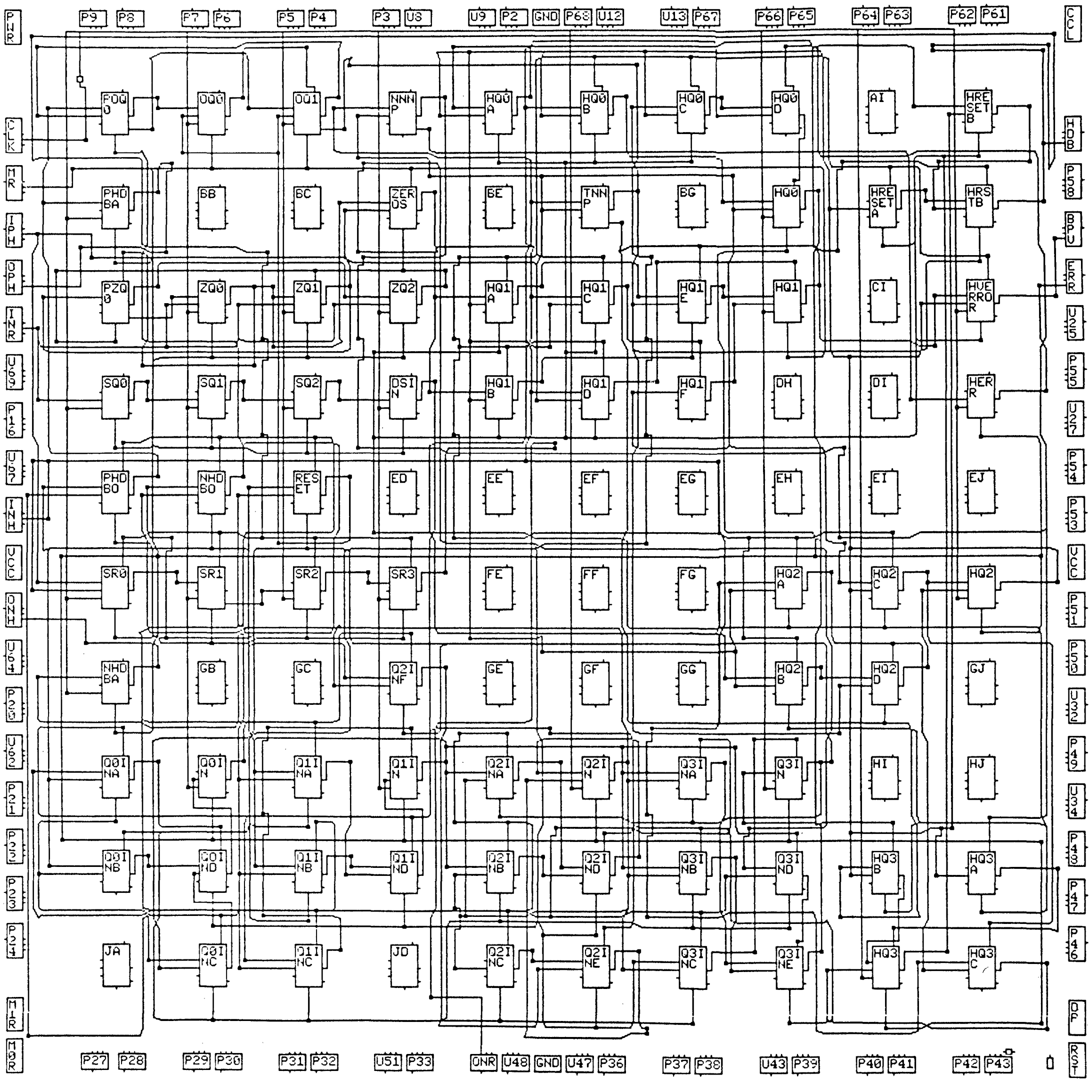


Figure 12. HDB3 Line Transcoder LCA Design

Logic Cell Array Design Methodology

All three designs are composed of similar building blocks, they all use shift registers, combinational logic and state machine logic.

Each of the three line transcoders require the use of two sets of shift registers. One set delays the input NRZ data, and the other set buffers the output NRZ data. To design a shift register using a CLB is simple. Two inputs (input variable, clock) are used to generate a single output (Figure 13); this output is then cascaded with additional registers to form a shift register. In the B8ZS design, five CLBs are used to implement a 5-bit shift register.

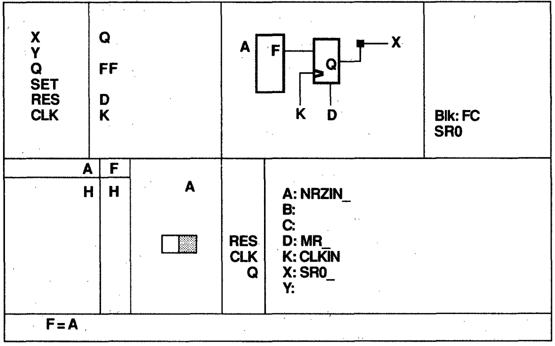


Figure13. Basic Element of N-bit Shift Register Configured in a Single CLB

For state machines and combinational logic, both the state machine equations and combinational logic equations must be carefully partitioned to fit them into CLBs and IOBs. Two forms of optimization can be done. One can either optimize for implementation efficiency i.e., minimize the usage of CLBs and IOBs, or, one can optimize for speed. To optimize for CLB usage, one must carefully analyze the equations and group signals which can potentially share the same CLB outputs. On the other hand, to optimize for speed, one must pay attention to the routing paths used as well as fanout from each CLB output and the way in which each CLB is configured.

For example, the HDB3 encoding state machine uses four state variables (Q3, Q2, Q1, and Q0) in order to generate fifteen states. First, the state equations are derived from the state diagram. Then, the state equation is partitioned to fit into CLBs. The following equation is the most significant state variable (Q3) of this HDB3 encoding state machine.

Six CLBs are used to implement this Q3 state equation. Five of the CLBs (Q3INA, Q3INB, Q3INC, Q3IND, and Q3INE) are configured as 4-input variable CLBs which generate a single output. Because these CLBs perform the preliminary logic for input into the block Q3IN, no storage element is necessary in these CLBs. Instead, the storage function is implemented in the block Q3IN. The design details for each CLB is shown in Figure 14.

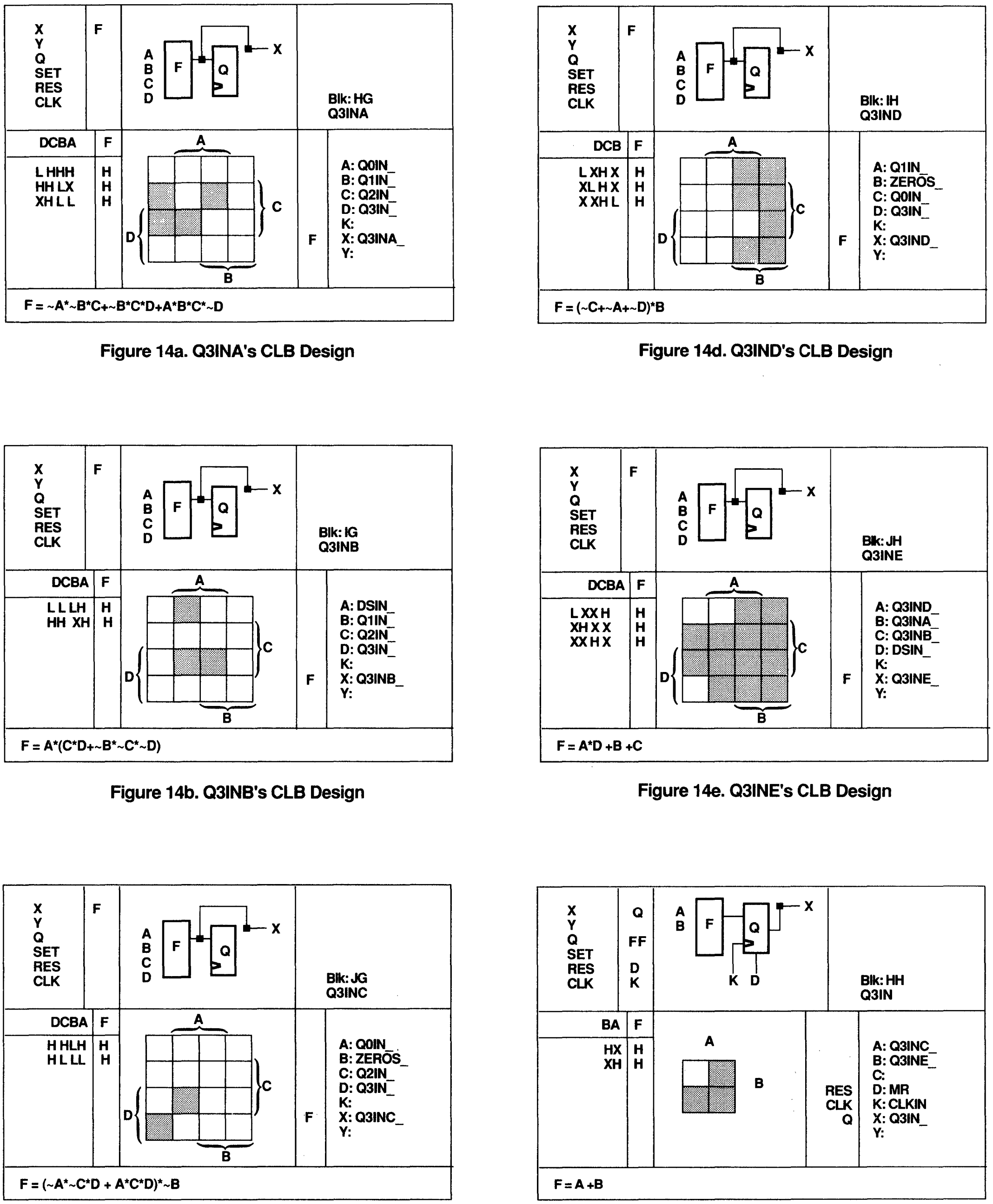
The B8ZS line transcoder can be implemented using three PAL® devices: two 16R8 devices and one 16R6 device. All equations for those PAL devices are assertive low; e.g., $\overline{C2}$:= C2 * C1 * C0 + $\overline{C2}$ * $\overline{C1}$ + $\overline{C2}$ * $\overline{C0}$ + RST + NRZIN. The LCA device can easily incorporate this assertive low equation simply by adding a tilde ("~") in front of the output equation (see Figure 15).

$$\begin{aligned}
Q3 := \; & \overline{Q0} * \overline{Q1} * Q2 && ;Q3INA\\
+ \; & \overline{Q1} * Q2 * Q3 &&\\
+ \; & Q0 * Q1 * Q2 * \overline{Q3} &&\\
+ \; & DSIN * Q2 * Q3 && ;Q3INB\\
+ \; & DSIN * \overline{Q1} * \overline{Q2} * \overline{Q3} &&\\
+ \; & \overline{Q0} * \overline{Q2} * Q3 * \overline{ZEROS} && ;Q3INC\\
+ \; & Q0 * Q2 * Q3 * \overline{ZEROS} &&\\
+ \; & DSIN * \quad \overline{Q0} * ZEROS &&\\
+ \; & DSIN * \quad \overline{Q1} * ZEROS && ;Q3IND\\
+ \; & DSIN * \quad \overline{Q3} * ZEROS &&\\
& ;Q3INE &&\\
& ;Q3IN &&
\end{aligned}$$

The HDB3 encoder has an ODD_EVEN state machine to check the number of preceding logic ones being odd number or even number. Its ODD_EVEN signal feeds back into the HDB3 encoding state machine. If these two state machines reference the same rising clock edge. It is possible to obtain the wrong result because the HDB3 encoding state machine always gets the ODD_EVEN signal one clock early. Using the falling clock edge can compensate for this timing problem. The positive and negative clock edges are available in each CLB. In this ODD_EVEN state machine, the negative clock is selected (see Figure 16).

Other common functions of communication equipment can be easily implemented using the reconfigurability of the LCA device. For example, two duplex schemes: half duplex and full duplex, can be implemented easily because of the reconfigurablilty of the IOB. Each IOB can be configured to an input, an output, or a bidirectional input/output with three-state control. Two IOBs are used to implement the full duplex communication, this scheme allows the device to transmit and receive at the same time (see Figure 17a). It can be easily modified to the half-duplex communication simply by using one bidirectional IOB, the information must be transmitted or received exclusively (see Figure 17b).

In addition, loop-back feature can be implemented by reconfiguring LCA's interconnect. Remote loop-back is implemented by disconnecting the line transcoder with the input and output pads, but connecting the input pad to the output pad directly within the transcoder (see Figure 18a). Local loop-back can also be implemented. Connecting the output of the transmitter to the input of the receiver directly, but disconnecting with input and output pads (see Figure 18b).



Blk: HG
Q3INA

DCBA	F
L HHH	H
HH LX	H
XH L L	H

A: Q0IN_
B: Q1IN_
C: Q2IN_
D: Q3IN_
K:
X: Q3INA_
Y:

F = ~A*~B*C+~B*C*D+A*B*C*~D

Figure 14a. Q3INA's CLB Design

Blk: IH
Q3IND

DCB	F
L XH X	H
XL H X	H
X XH L	H

A: Q1IN_
B: ZEROS_
C: Q0IN_
D: Q3IN_
K:
X: Q3IND_
Y:

F = (~C+~A+~D)*B

Figure 14d. Q3IND's CLB Design

Blk: IG
Q3INB

DCBA	F
L L LH	H
HH XH	H

A: DSIN_
B: Q1IN_
C: Q2IN_
D: Q3IN_
K:
X: Q3INB_
Y:

F = A*(C*D+~B*~C*~D)

Figure 14b. Q3INB's CLB Design

Blk: JH
Q3INE

DCBA	F
L XX H	H
XH X X	H
XX H X	H

A: Q3IND_
B: Q3INA_
C: Q3INB_
D: DSIN_
K:
X: Q3INE_
Y:

F = A*D +B +C

Figure 14e. Q3INE's CLB Design

Blk: JG
Q3INC

DCBA	F
H HLH	H
H L LL	H

A: Q0IN_
B: ZEROS_
C: Q2IN_
D: Q3IN_
K:
X: Q3INC_
Y:

F = (~A*~C*D + A*C*D)*~B

Figure 14c. Q3INC's CLB Design

Blk: HH
Q3IN

BA	F
HX	H
XH	H

RES
CLK
Q

A: Q3INC_
B: Q3INE_
C:
D: MR
K: CLKIN
X: Q3IN_
Y:

F = A +B

Figure 14f. Q3IN's CLB Design

Figure 14. The CLB's Design for the Q3 Equation

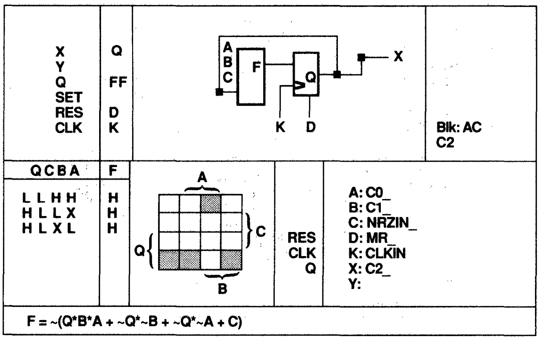


Figure-15. C2 is Represented by ~(C2)

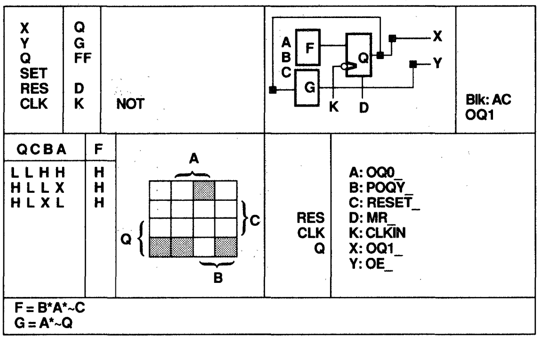


Figure 16. Negative Clock Edge in the CLB

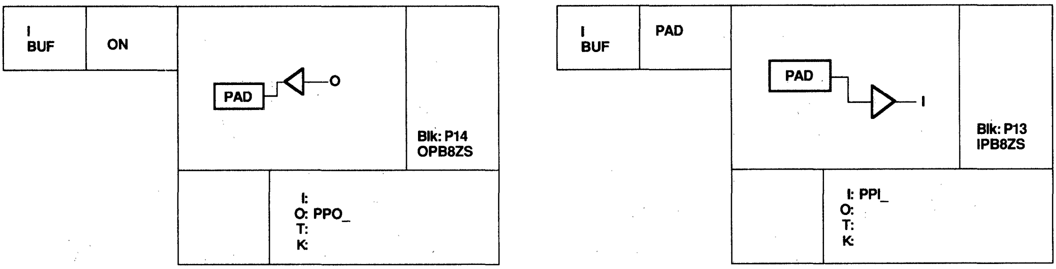


Figure 17a. Full-Duplex Communication

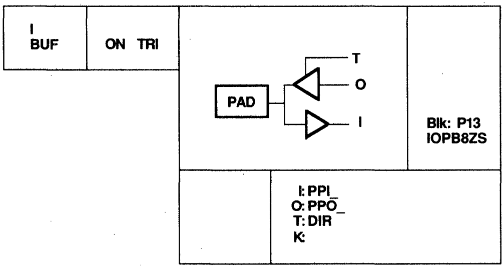


Figure 17b. Half-Duplex Communication

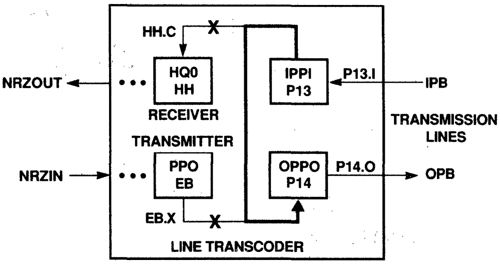


Figure 18a. Remote Loop Back

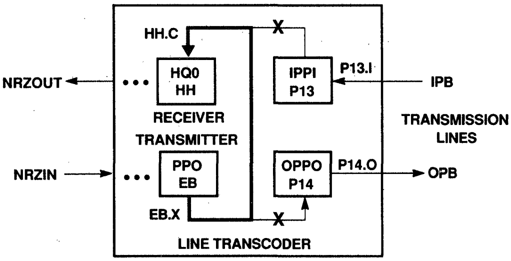


Figure 18b. Local Loop Back

Conclusion

This design demonstrates the power versatility and benefit of using the LCA device in line coding by exploiting the reconfigurable feature of the device. A single M2018 LCA device can support three different line codes, such as B8ZS, B6ZS, and HDB3 codes.

The B8ZS is used at either 1.544 Mbps T1 carrier or 6.312 Mbps T2 carrier on coaxial pairs. The B8ZS encoder uses twenty-seven CLBs to convert NRZ data into two B8ZS signals, and the B8ZS decoder uses thirty-nine CLBs to convert two B8ZS signals to NRZ data. The B6ZS is used at 6.312 Mbps T2 carrier on symmetrical pairs. The B6ZS encoder uses twenty-nine CLBs to convert NRZ data into two B6ZS signals, and the B6ZS decoder uses thirty-eight CLBs to convert two B6ZS signals to NRZ data. The HDB3 is used at 2.048 Mbps or 8.448 Mbps transmission lines. The HDB3 encoder uses forty CLBs to convert NRZ data into two HDB3 signals and the HDB3 decoder uses thirty-two CLBs to convert two HDB3 signals to NRZ data.

The line transcoder is required when the signal is transmitted over transmission lines. It converts unipolar binary codes to bipolar codes or modified bipolar codes in order to eliminate DC wander and provide the timing information.

References

1. Theresa Shafer and Steve Patterson, *"B8ZS Coding,"* Monolithic Memories Application Note AN-169.
2. Cindy Lee, *"HDB3 Line Coding using Three PAL® Devices,"* Monolithic Memories Application Note AN-176.
3. John C. Bellamy, *"Digital Telephony,"* John Wiley & Son, 1982.
4. Recommendation G.703, CCITT Red Book, Volume III.

The detailed LCA design files are available from Advanced Micro Devices.

The design file of B8ZS refers to XDES12.LCA.

The design file of B6ZS refers to XDES13.LCA.

The design file of HDB3 refers to XDES14.LCA.

Building an ESDI Translator Using the M2064 Logic Cell™ Array

Ken Won and Ken Tseng

Abstract

The ESDI (Enhanced Small Device Interface) Standard is a low-cost, high-performance Winchester disk drive interface. The ESDI Translator is an interface controller that is implemented in an ESDI-compatible disk drive.

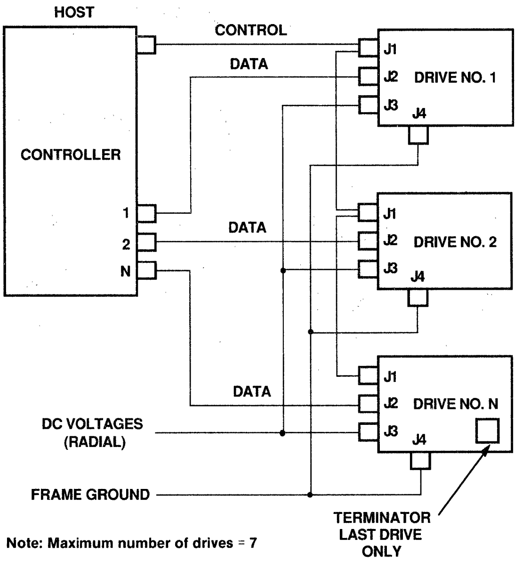
The ESDI Translator is implemented in one M2064 LCA device. The LCA device is a RAM-based high-density CMOS integrated circuit which is reprogrammable. This application note describes the design implementation and design considerations of the ESDI Translator on the LCA device.

Building an ESDI Translator Using the M2064 Logic Cell Array

AN-179

The ESDI Translator

ESDI is a low-cost, high-performance interface standard suitable for the smaller, high-performance Winchester disk drives currently being produced. The ESDI interface consists of a control cable and a data cable. The control cable allows for a daisy chain connection of up to seven devices (disk or optical drives) with only the last device being terminated. In our design, we assumed that the device is a disk drive. The data cable is attached in a radial configuration (See Figure 1).



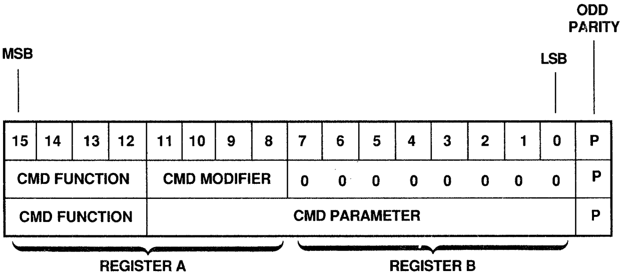
Note: Maximum number of drives = 7

Figure 1. Connection Between the Controller and Multiple Drives

The ESDI Translator handshakes serial commands from a disk controller, deserializes the commands and passes the commands to a microcontroller. The command data word structure is shown in Figure 2.

The Command Function bits define functions to be executed by the disk drive. These functions are seek, recalibrate, request status, request configuration, select head group, control, data strobe offset, track offset, initiate diagnostics, set bytes per sector and set configuration. Some of these functions, such as the control function, have modifiers for more detailed functional description. Other commands have parameters that contain numbers. For the seek command, the parameter specifies the cylinder number that the drive will seek to. The request status and request configuration commands require data from the disk drive to be transferred back to the disk controller. In our current design, internal registers A and B in the LCA device represent the upper and lower bytes of the command respectively.

Figure 3 illustrates the relationships between the disk controller, the ESDI Translator, and the microcontroller. The PROM is used to store the configuration data for the LCA device. The Done/$\overline{\text{Program}}$ (D/$\overline{\text{P}}$) output is driven LOW when the device is being configured. Configuration data is read from the PROM device during configuration. After configuration is complete, the LCA device drives the D/P pin HIGH to deselect the PROM. Drive selection, write protection, command completion and fault detection are also handled by the ESDI Translator.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	P
CMD FUNCTION				CMD MODIFIER				0	0	0	0	0	0	0	0	P
CMD FUNCTION				CMD PARAMETER												P

Figure 2. Command Data Word Structure

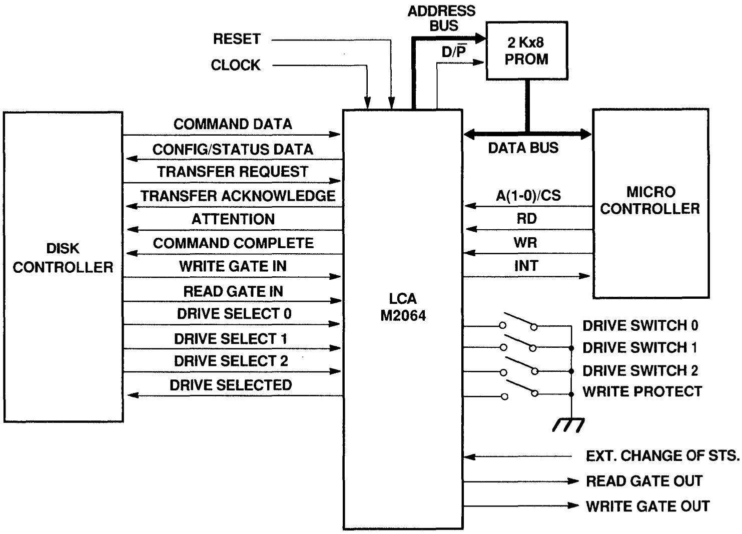


Figure 3. An ESDI Translator Implemented on the LCA Device

Why Use an LCA Device in an ESDI Translator

The ESDI interface standard requires more logic functions to be built into the disk drive than some other interface standards, such as the ST506 standard. However, the external dimensions of a disk drive usually have to conform to an industrial standard form factor. Thus, the use of high-density semicustom chips is the logical solution to increase functionality without increasing the external dimensions.

The LCA device is a high-density CMOS integrated circuit available from Monolithic Memories. Its high gate density allows the implementation of an ESDI Translator in a single chip. Fifteen standard SSI and MSI chips would be necessary for the same application. If PLDs are used to implement the ESDI Translator, more than one would be necessary because a large amount of logic is required, thus occupying more board space.

A major advantage in using the LCA device is that it can speed up the design cycle, enabling the manufacturer to have a shorter time-to-market. Also, many peripheral products are produced in relatively small quantities aiming at very specialized markets. The LCA device, which has no NRE cost, makes the production of small quantities more economical than the gate array. Another advantage of the LCA device over other semicustom chips, such as the gate array, is its reprogrammability feature. The LCA device is RAM-based which can easily be reprogrammed by the user in the final system. This feature is especially important in the peripheral products market, where many products have short life spans.

Design Implementation

The ESDI Translator is responsible for all control interfaces between the disk controller and the disk drive. An internal block diagram of the ESDI Translator is shown in Figure 4. It consists of five major logic building blocks:

- Drive selection
- Read gate/write gate
- Counter/controller
- Shift register and parity generator/checker
- Internal register address decoder

Drive selection on ESDI-compatible drives involves three signals from the disk controller, Drive Select 0-2. These three drive select lines are encoded so that up to seven drives may be connected to the same ESDI port, as shown in Table 1.

On the LCA device, the drive number is selected by connecting the drive switch pins to either VCC or GND. When the code on the drive select lines, DSX, equals the code on the drive switch pins, DSWX, where X may be 0, 1, or 2, the drive is selected and the Drive Selected signal, DSELD, is asserted. Once the drive has been selected, serial commands output by the disk controller will be read by the LCA device. The actual implementation is shown in Figure 5, using two CLBs and seven IOBs. BDS0 and BDS1 are the names of the CLBs in the current design. In our design, names that begin with the letter B or P are used to designate a CLB or an IOB respectively. SCLK is the system clock.

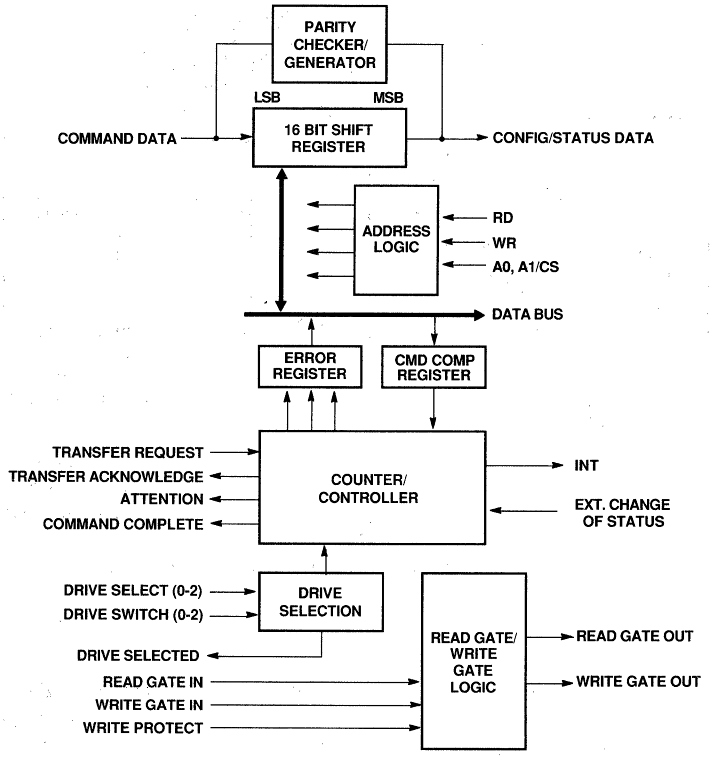


Figure 4. ESDI Translator Internal Block Diagram

DS2/ DSW2	DS1/ DSW1	DS0/ DSW0	DRIVE
0	0	0	None
0	0	1	Select Drive 1
0	1	0	Select Drive 2
0	1	1	Select Drive 3
1	0	0	Select Drive 4
1	0	1	Select Drive 5
1	1	0	Select Drive 6
1	1	1	Select Drive 7

Table 1. Drive Selection

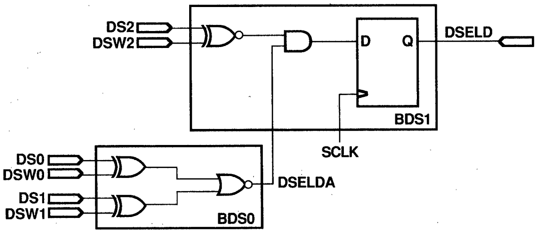


Figure 5. Configuration of the BDS0 and BDS1 CLBs to Provide the Drive Selected Signal

Figure 6 shows the configurations of the two CLBs as displayed on the computer screen by the XACT development software. In Figure 6a, the CLB is configured as one function of four variables. The D flip-flop is not used. The logic representation, truth table, Karnaugh map, signal names, block name and Boolean equation are shown.

In Figure 6b, the CLB is configured as one function of three variables, the output of which is connected to the D flip-flop. The Q output of the D flip-flop becomes the output of this CLB.

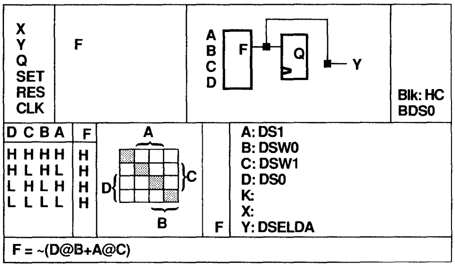


Figure 6a. Drive Selection Logic Implemented in a CLB (BDS0)

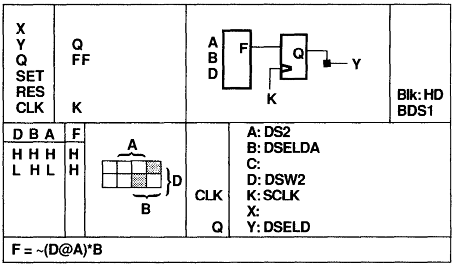


Figure 6b. Drive Selection Logic Implemented in a CLB (BDS1)

The LCA device also performs logic functions for the Read Gate and Write Gate logic blocks. The Read Gate signal allows data to be read from the disk, and the Write Gate signal allows data to be written on the disk. These signals from the disk controller are input to the LCA device as Read Gate In and Write Gate In. Under normal operating conditions, Read Gate Out is asserted when Read Gate In is asserted and Write Gate Out is asserted when Write Gate In is asserted. When both Read Gate In and Write Gate In are asserted, a write error condition results and the Attention line is asserted, signalling the disk controller that an error has occurred. Also, the LCA device may be used to provide write protection to a disk drive. When the Write Protect signal is asserted, the Write Gate Out signal will not be asserted when the Write Gate In signal is asserted, preventing the write circuitry from being activated. These logic functions are implemented in two CLBs, BRWG0 and BRWG1, as shown in Figure 7.

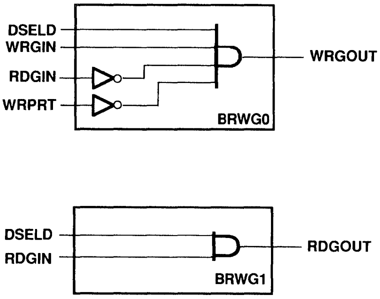


Figure 7. Read Gate/Write Gate Logic Implementation

The Counter/Controller handshakes commands from the disk controller. It also handshakes status/configuration data to the disk controller. It is also responsible for generating the Interrupt, Attention and Command Complete signals. Seventeen bit commands (one bit is parity) are transferred from the disk controller to the LCA device via the Command Data line. The serial bit transfer is performed using a pair of handshaking signals, Transfer Request (TREQ) and Transfer Acknowledge (TACK). TREQ is asserted by the disk controller when a bit is valid on the Command Data line, and TACK is asserted by the LCA device when the command bit has been read. The handshaking action is shown in Figure 8.

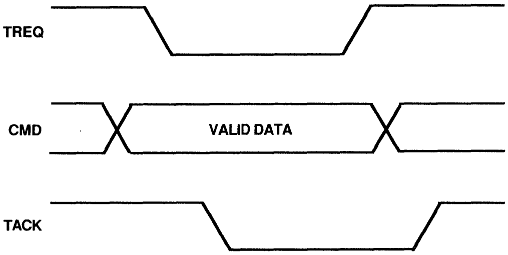


Figure 8. Command Data Transfer

A 17-state counter counts the number of command bits shifted in or shifted out of the data registers. Its implementation is shown in Figure 9. The SHIFT signal is asserted, thus incrementing the counter, whenever there is a transfer request and the LCA device is selected. CNT_Q0 is the lowest bit of the shift register counter. This bit is inverted whenever SHIFT is asserted unless sixteen bits have already been shifted into the LCA device (CNT16 asserted, or CKCMD asserted when all seventeen bits have been shifted in). The DATAOUT signal is asserted after a Request Status command or a Request Configuration command has been transferred and the internal data registers have been loaded with data to be serially shifted out. Hence, when DATAOUT is asserted, CKCMD is negated and the counter is enabled.

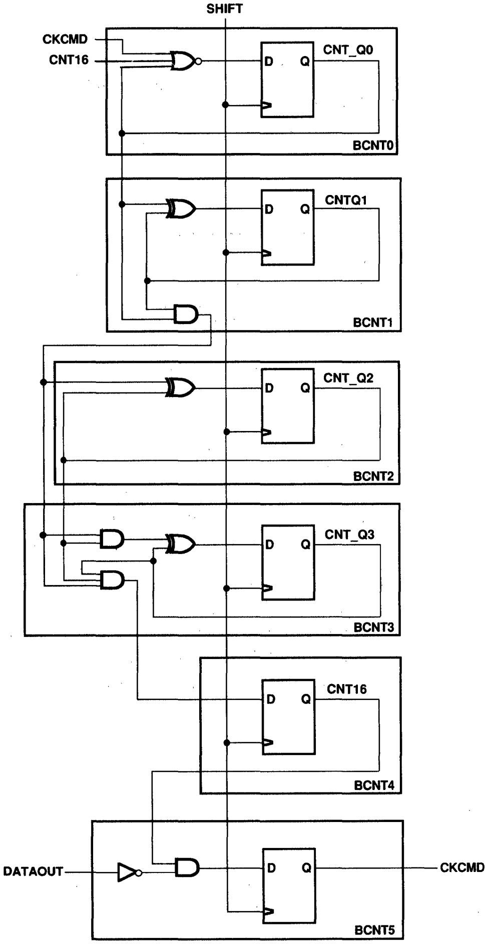


Figure 9. 17-State Counter Implemented in six CLBs

The logic generation of the Attention signal, ATTEN, is shown in Figure 10. Whenever there is a write fault, WRFLT, a parity error, PARERR, an interface fault, INTFLT, or an external error, CSTS(Change of Status), the ATTEN signal is asserted. When both WRGIN and RDGIN signals are asserted and the LCA device is selected, the WRFLT signal is asserted. The WRFLT signal is negated when the command transferred to the LCA device Register A is the Reset command defined by the ESDI standard, which has a Command Function of 0101 (Control) and a Command Modifier of 0000 (Reset Attention and Standard Status). The Command Function and Command Modifier formats are shown in Figure 2.

Three CLBs, BPG0, BPG1, and BPG2, are responsible for generating the PARERR signal. BPG0 is a multiplexer that selects the source of the input to the parity generator/checker (BPG1). If data is being shifted into the LCA device (parity checker mode), the CMDBITA input is chosen. If data is being shifted out of the LCA device (parity generator mode), the RA_Q7 output is chosen. These two signals are also shown in Figure 11. BPG1 is an odd parity generator/checker. In the parity checker mode, whenever an odd number of ones are passed through this CLB, the output is one. This output signal is connected to BPG2, which inverts the signal and asserts or negates the PARERR signal accordingly. If parity is correct, an interrupt is asserted to the microcontroller informing the microcontroller that a command has been received and is ready to be read. In the parity generator mode, the output of BPG1 is the parity bit. BPG1 is clocked by the SHIFT input, which is asserted whenever there is a transfer request and the LCA device is selected. BPG1 is reset by the Reset Parity Generator input, RSTPGEN. This signal is asserted when either INT is asserted or an interface fault is detected.

The Interface Fault signal is asserted when the LCA device is selected and CNTR is negated before seventeen bits have been transferred. CNTR is asserted after the first command bit is shifted in and is negated after the seventeenth bit is shifted out. BINTFLTO is clocked by the system clock, SCLK, and reset by the RSTCOS signal. The IOB PCSTS is configured as a buffered input to signal external error conditions.

The microcontroller is able to address four register locations in the LCA: two data registers, BRGA and BRGB, one error register, BDMX, and one command complete register, BRC7. Only three bits in the error register are used. They are bit 0 for parity error, bit 4 for interface fault and bit 7 for write fault. Only one bit in the command complete register is used. This is bit 0, which has a value of zero when the command is completed. The addresses and contents of the registers are shown in Table 2.

A1	A0	REGISTER	BIT NUMBER							
			7	6	5	4	3	2	1	0
0	0	REGISTER A								
0	1	REGISTER B								
1	0	ERROR REGISTER		▨	▨		▨	▨	▨	
1	1	COMMAND COMPLETE REGISTER	▨	▨	▨	▨	▨	▨	▨	

Table 2. Register Addresses and Contents

The LCA device implementation of the registers and multiplexer is shown in Figure 11. When the interrupt signal is asserted, the microcontroller reads the two data registers by setting the A1 and A0 address lines appropriately and asserting the RD signal to the LCA device. These data registers contain the command that is transferred from the disk controller through the CMDBITA input. If the command is a request data command, configuration or status data is written to these two data registers by the microcontroller. These two bytes, plus a parity bit that is generated by the parity generator in the LCA device, are serially transferred to the disk controller over the Config/Status Data line through RA_Q7. After all seventeen bits have been transferred, the Command Complete signal is asserted. If the command is not a request data command, the microcontroller executes the command and upon completion, writes a byte of zeros to the command complete register of the LCA. When the command complete register is written with all zeros, the Command Complete signal is asserted by the LCA device.The command complete register may only be written, not read. The status register contains error bits that are set when errors are detected. This register may only be read and not written.

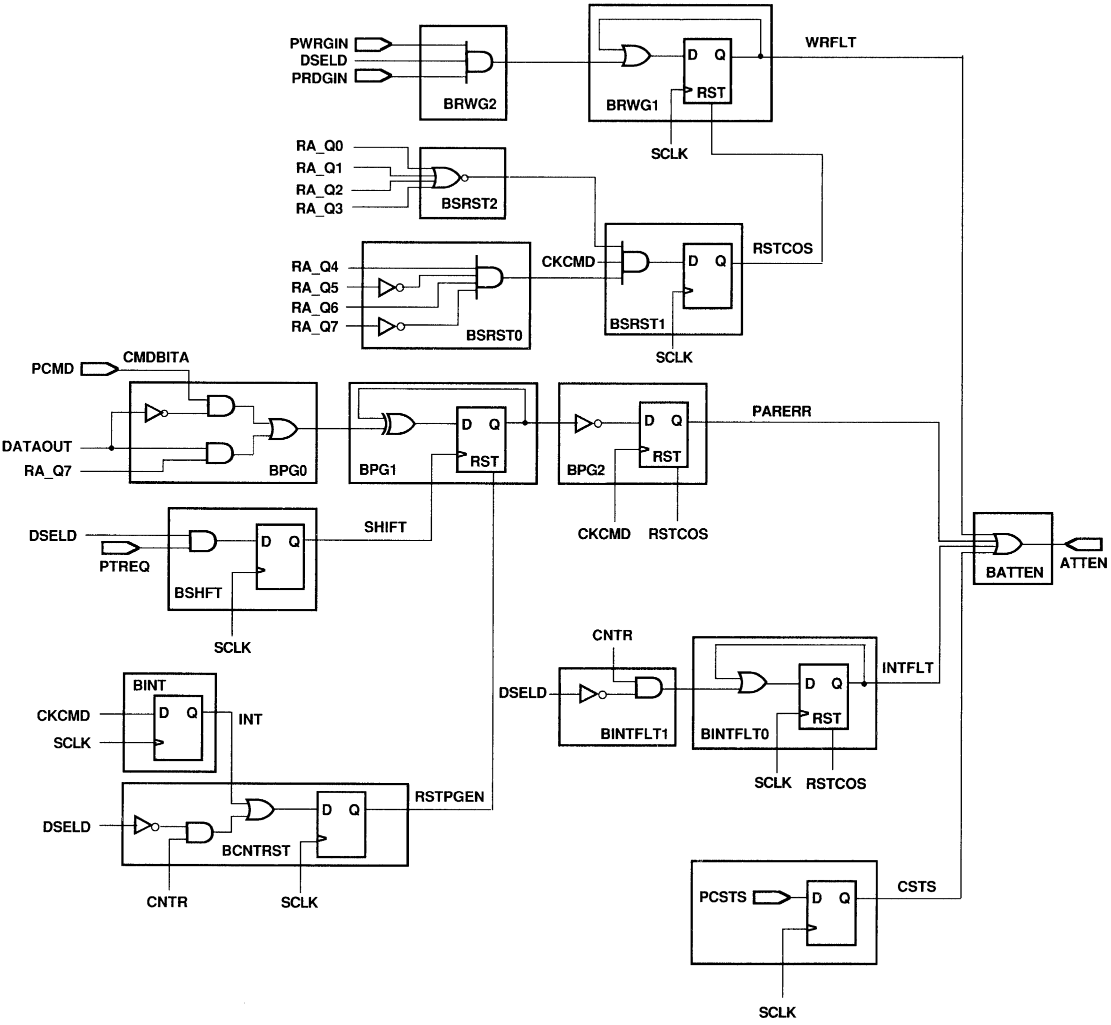


Figure 10. Generation of the Attention Signal by the Four Possible Error Conditions

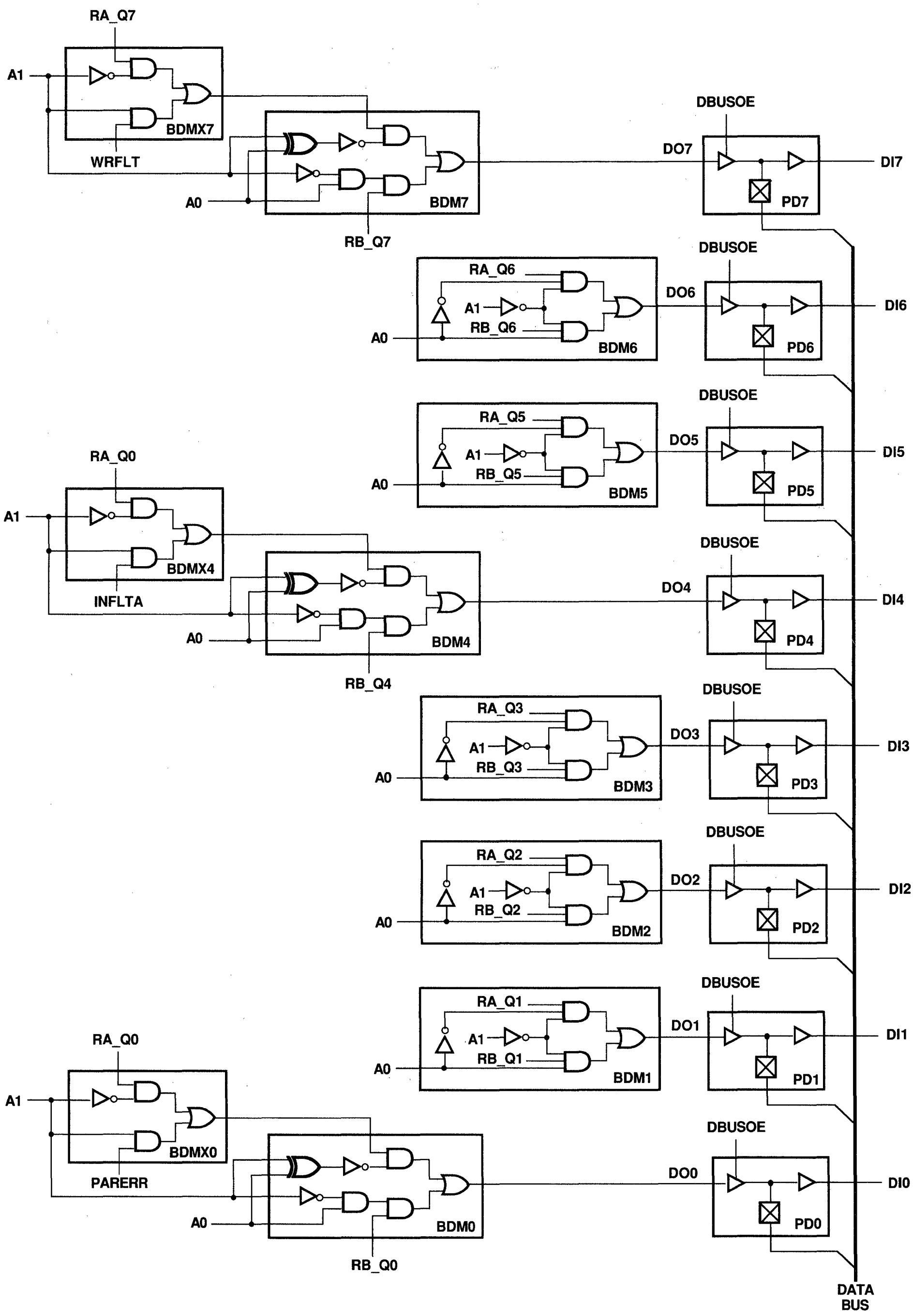


Figure 11. Registers A, B, Error Register, and Multiplexer

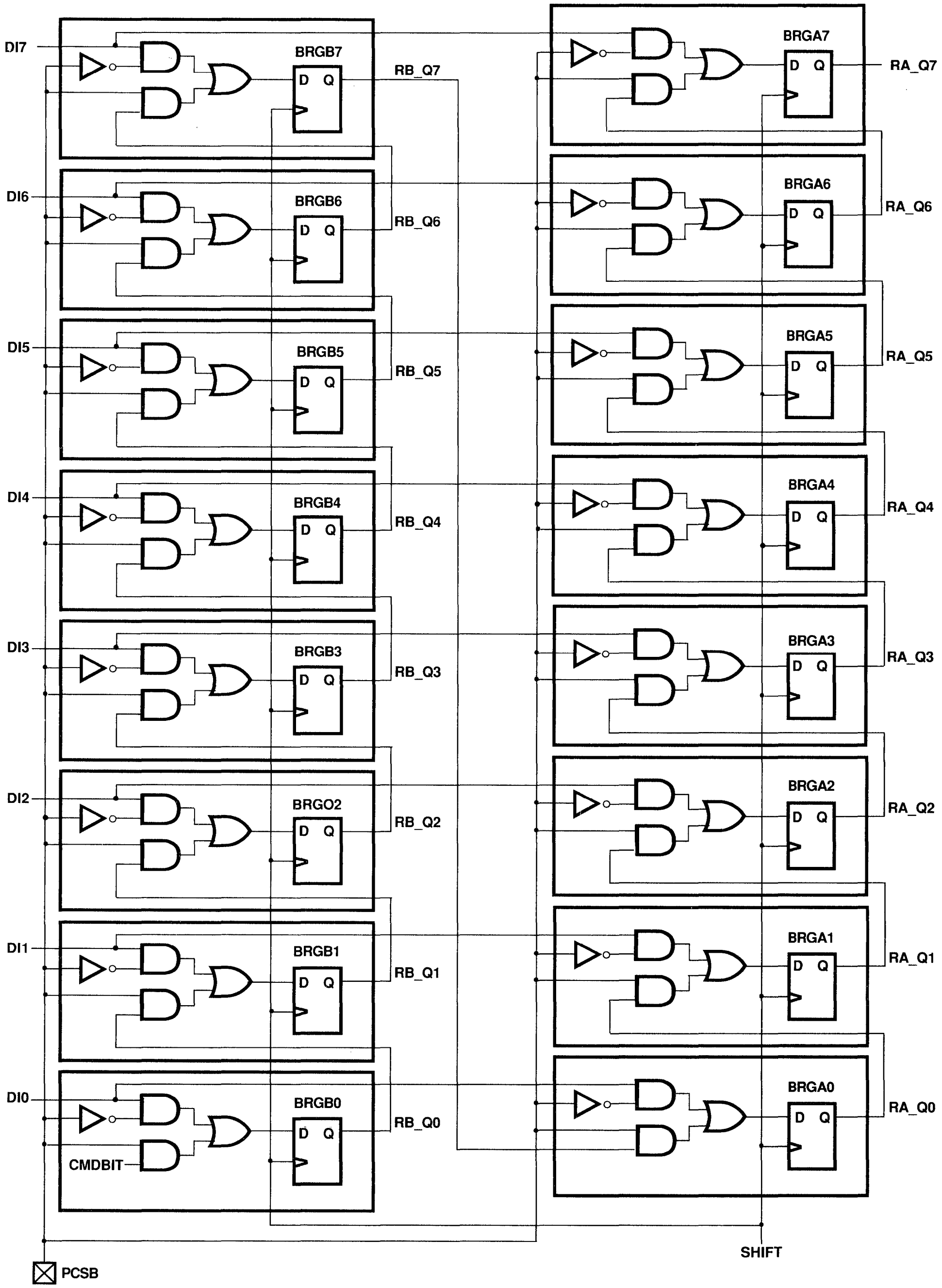


Figure 11. Registers A, B, Error Register, and Multiplexer (Continued)

Design Considerations

The circuit diagram of the programmed LCA device is shown in Figure 12. This design implementation uses sixty-three of the sixty-four CLBs within the LCA. This translates into a 98% usage. The rightmost column of CLBs contains the multiplexer, which selects between registers A, B, or the error register to be placed on the data bus. Registers A and B are placed on the third and second column from the right end, respectively. The registers and multiplexer are placed physically close to each other to simplify routing. The CLBs which execute a particular function are placed physically next to each other. These functions include the parity generator/checker, 17-state counter, read gate/write gate logic, drive selection logic and error detection logic. The read gate/write gate logic and the drive selection logic, which require much I/O activity, are placed in CLBs near the edges of the device.

Another consideration in implementing an LCA design is routing. Long lines are available for signals that have to travel a long distance within the LCA device. These lines can also be used for signals that must have minimal skew between different destinations. An automatic routing program is available in the XACT software package.

Routing requires careful consideration when a design uses a high percentage of the available CLBs (above 95%). Although automatic routing is available, designs with high CLB usage may require point-to-point routing (EDITNET command in XACT). Some of the techniques that can be used are swapping input pins, swapping CLBs, and implementing buffers in unused sections of the CLBs. When swapping input pins, the designer must be careful because certain functions, such as the clock input to the flip-flop, may only be input on certain pins. Swapping CLBs is very simple because the XACT software provides a command for swapping CLBs, but sometimes not all of the signals can be successfully rerouted. Passing a signal through a CLB presents a routing channel that is not otherwise available. However, a delay is added to the signal. Usually, a combination of these techniques can be used to successfully route a high-density design.

The high number of IOBs in the LCA device gives the designer much flexibility in designing the pinout of the device. In this design, thirty-five of the IOBs are used. The IOBs that are not being used for the actual design are not left unused, but are configured as outputs and are connected to various signals within the LCA device to provide test points for the LCA device. This greatly increases the testability of the design once it is placed on the board.

XACT is used by the designer to define the CLBs and IOBs, and to perform EDITNET. Alternately, the design may be input using the schematic capture software offered by Daisy and Futurenet. In these methods, an automatic place and route software package divides the design into blocks that may be implemented in CLBs and IOBs, and routes the CLBs and IOBs automatically. Once the design has been completed, it may be simulated using the software package P-SILOS.

Conclusion

The LCA device provides many advantages to the user. Its high gate count and I/O capability could potentially replace several PLDs in many applications, hence reducing board space. The LCA device is preferred by many customers over gate arrays because it is reprogrammable 'on-the-fly' and there are no long design cycles and initial NRE cost of a gate array. The current design file, XDES15.LCA, is available upon request. The bit pattern and the .LCA file will be provided for programming the LCA device in an EPROM.

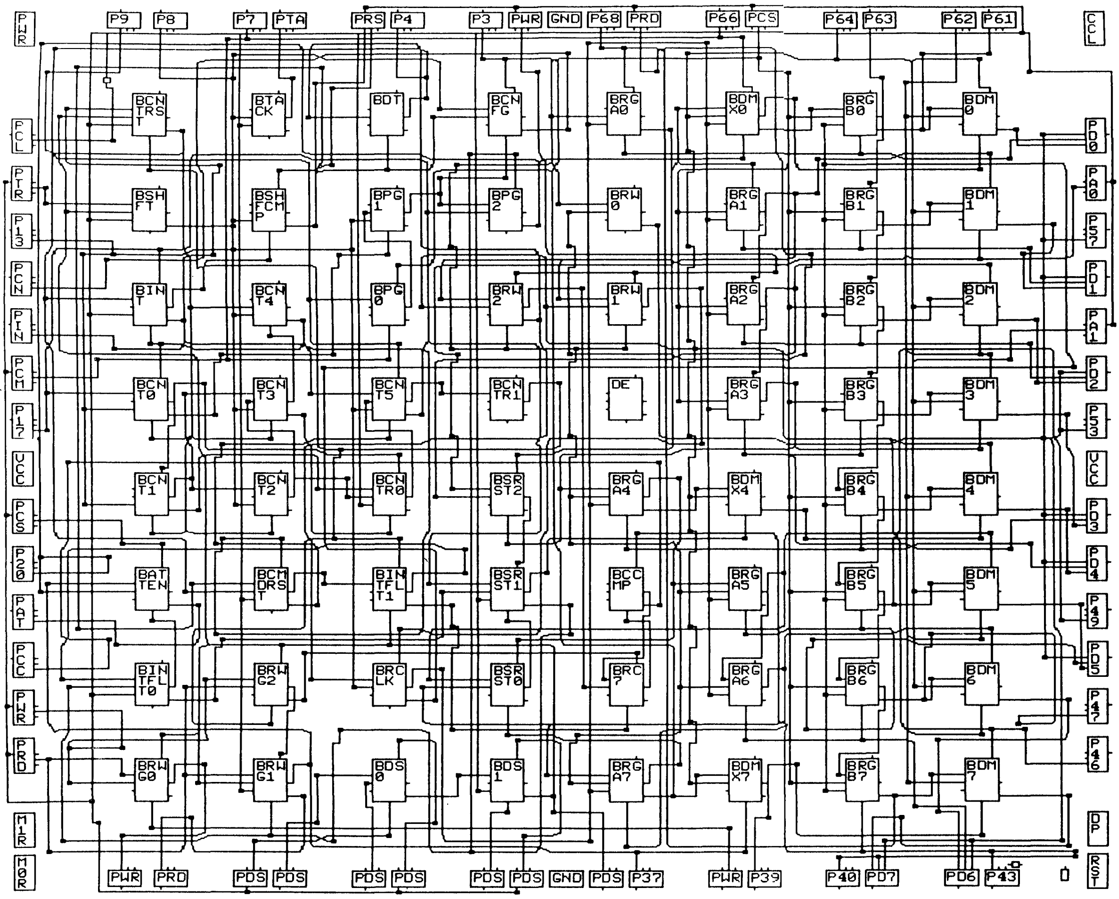


Figure 12. ESDI Translator LCA Design

Advanced
Micro
Devices

AN-184

Using the Logic Cell™Array to Build a Pseudo-Random-Number Generator

Mohammed Wasfi

Abstract

The Logic Cell Array is a programmable integrated CMOS circuit that can easily be configured to perform many LSI functions. This Application Note will discuss how the LCA device can be used in the design of an 8-bit Pseudo-Random-Number Generator. The design consists of three major components: the registers that generate the random numbers, the 1-Hz Clock, and the decoding circuitry. The decoding circuitry can be used with two seven-segment displays to show the value of the binary random number in hexadecimal (00-FF) format.

10288A
MARCH 1988

Using the Logic Cell Array to Build a Pseudo-Random-Number Generator

Mohammed Wasfi

Introduction

The LCA device is a high-density programmable integrated circuit. Due to its flexible architecture and programmability, it can be configured to perform many functions. In this case, the LCA device was configured as an 8-bit Pseudo-Random-Number (PRN) Generator. The high density of this device made it possible to implement additional needed circuits such as the registers, clock, and the decoders for the seven-segment displays. Additional information on the LCA device can be found in the *LCA Applications Handbook.* Information on configuring the LCA device is discussed in Application Note 182.

Random Number Applications

A Pseudo-Random-Number Generator can be used in a wide range of applications. A common application for the PRN Generator is electronic games where a random number is used to create a random event. Another sophisticated application is signal scrambling, for securing communications, where a random number is added to a transmitted signal for scrambling, and subtracted for descrambling.

Pseudo-Random-Number (PRN) Generation

There are several different methods of generating a random number. In this LCA device application, however, the PRN is generated using a shift register with feedback. To achieve a random output, the feedback comes from the prime number bits (flip-flops).

Although the output of the shift register is a random number, the cycle of random numbers generated is repeated after a certain period. This is indicated by the term " pseudo." The length of the random number generation period depends on the number of flip-flops used. The period length is calculated as follows:

Period = 2^n-1, where n is the number of flip-flops.

The reason for the subtraction of one state is that the state of all flip-flops set to zero is illegal (causes a lockout).

The Logic Cell Array

The LCA device is a configurable large scale integrated CMOS circuit. It is RAM based, so its configuration can be changed by simply loading it with a new configuration bit stream. The device architecture consists of Configurable Logic Blocks (CLBs) and configurable I/O Blocks (IOBs). Each CLB has a register and can implement any one function of up to four variables or any two functions of up to three variables. The IOB can be configured as an input, registered input, output, or an output with a three-state function.

Two types of LCA devices are currently available from Monolithic Memories, the M2064 and the M2018. The M2064 has 64 CLBs arranged as an 8 by 8 matrix. The M2018 has 100 CLBs arranged as a 10 by 10 matrix. For this design, an M2064 was used because only 52 CLBs were needed to implement the design fully (Figure 1).

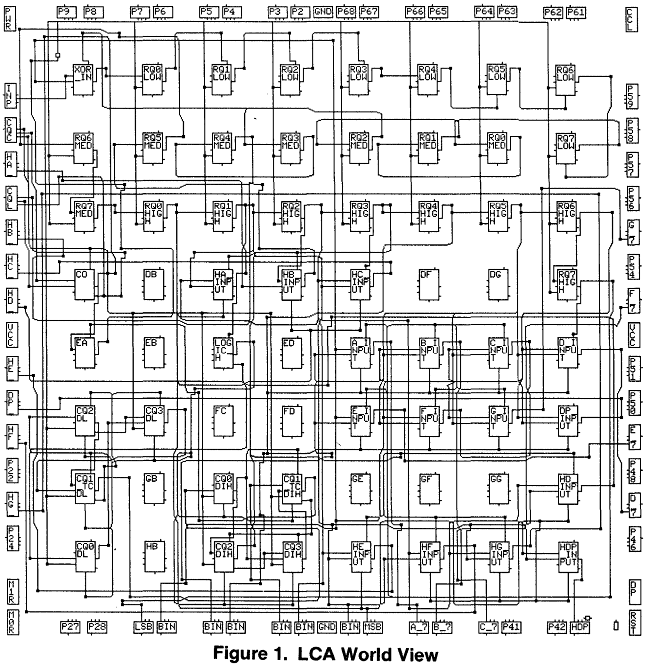


Figure 1. LCA World View

Why Use an LCA Device?

There are several reasons for choosing the LCA device to design a random number generator. The LCA device not only offers a large number of flip-flops in the CLBs, but also offers additional IOB flip-flops (one flip-flop per CLB/IOB, one hundred CLBs and seventy-four IOBs in the M2018, sixty-four CLBs and fifty-eight IOBs in the M2064). This enables the user to generate a very large period during which the PRN sequence will not repeat itself. In addition, the large number of configurable IOBs available makes it easy to generate a large random number.

One of the many features of the LCA device is its high density. There are approximately 1,200 gates in the M2064 and 1,800 gates in the M2018. The high gate count allows the implementation of additional circuits, such as the clock for the registers that normally are added outside of the chip.

Another feature of the LCA device is its configurability. For example, the ability to output a PRN of a certain size, then change the size to make it smaller or larger, or change other logic without having to use a new part is a big advantage.

With an LCA device, one has access to a macro library in the XACT™ Development System, which is the software used to configure the LCA device (available from Monolithic Memories). These macros simplify the designs considerably. Approximately 75% of this design used Monolithic Memories-supplied macros.

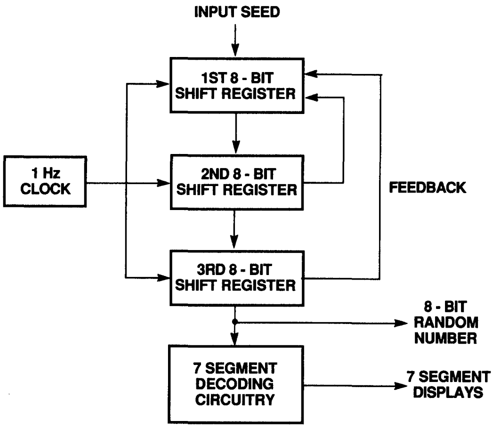

INPUT SEED
1ST 8 - BIT SHIFT REGISTER
1 Hz CLOCK
2ND 8 - BIT SHIFT REGISTER
FEEDBACK
3RD 8 - BIT SHIFT REGISTER
8 - BIT RANDOM NUMBER
7 SEGMENT DECODING CIRCUITRY
7 SEGMENT DISPLAYS

Figure 2. Block Diagram

Design Breakdown

The design of the PRN generator consists of three major components (Figure 2). The first component is the shift register. Three 8-bit shift registers are used to generate the PRNs using D-type flip flops. Since there are 24 flip-flops, the length of the PRN sequence period is:

2^{24}-1 = 16,777,215 states,

after which the PRN sequence repeats itself.

The second component is the clock for the flip-flops. A low frequency (100-HZ) R-C Oscillator was used. This frequency is then divided by 100 (using two modulo-10 counters) which results in a 1-Hz clock that is used for the flip-flops.

The third component is the decoding circuitry for the seven-segment displays. Two seven-segment display decoding circuits are implemented in the LCA device to show the hexadecimal equivalent of the binary output. This makes it easier to study the nature of the PRNs generated.

Design Implementation in the LCA Device

The main component in this design is the shift register. The design of a shift register is relatively simple. A register (flip-flop) in a CLB uses one input, a clock, and outputs the value that was the state of the input delayed by one clock period (Figure 5). Then all twenty-four registers in the CLBs are cascaded together forming a 24-bit shift register. The macro, RS8, available with the Monolithic Memories XACT Macrocell Library, is an 8-bit shift register (Figure 3). Therefore, it is possible to invoke that macro and cascade three 8-bit shift registers (RQL, RQM and RQH) to get the required shift register size.

The feedback from the shift register, in order to produce a random outcome, has to come from a prime number bit (register). Therefore, the feedback from bits 11 and 23 were chosen. Since the PRN sequence cannot start without a logic high input seed from an IOB configured as an input (Figure 7b), a certain function (XOR_F) implementing the input seed and the two feedbacks was used. This function guarantees lockout will not occur (such as in the case of the input staying at a logic HIGH), and also adds to the randomness of registered outputs (Figure 6).

XOR_F = (B + C) @ A

B : input seed
C : 2nd feedback (bit 23)
A : 1st feedback (bit 11)
@ : exclusive or function

The 8-bit output of the third shift register (RQH) is connected to eight (8) IOBs configured as outputs (Figure 7a). The output of RQH is also connected to the two seven-segment decoders.

The clock for the registers was chosen as 1 Hz. This frequency was generated using a low-frequency (100-Hz) resistor-capacitor oscillator. This oscillator is also available as a macro, GOSC, with the XACT Development System (Figure 8a). The values for the resistors and capacitors (R1, R2, C1, C2) used with CQ and CQL are 100 Kohms and 0.1 microfarads. The Boolean equations for the Clock are:

Q = ~R * (S+R)
QL = ~Q

A modulo-100 counter was used as a " divide by" to get the 1 HZ clock needed. The modulo-100 is implemented using two cascaded 4-bit BCD (modulo-10) counters. The macro for the counters is also available with the XACT Development System (Figure 8b). The output of the modulo-100 is routed to the Global Clock Buffer (GCB). The output of the GCB is routed to the registers via long-line interconnects.

Finally, two decoders for the seven-segment displays were designed. The least and the most significant four bits of the PRN used one decoder each to decode binary 0000-1111 as hexadecimal 0-F (Table I). The decoder outputs were routed to IOBs configured as output buffers, where they were wired to a bus driver (74LS244), that could be used to drive the seven-segment display. Eight CLBs per decoder were used (Figure 4). Each CLB was configured to control one segment in the display, including the Decimal Point (DP). See Table II.

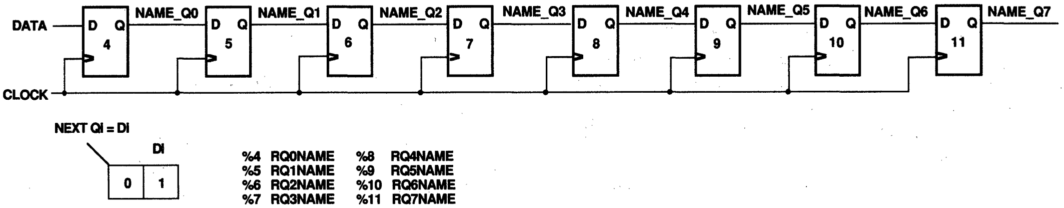
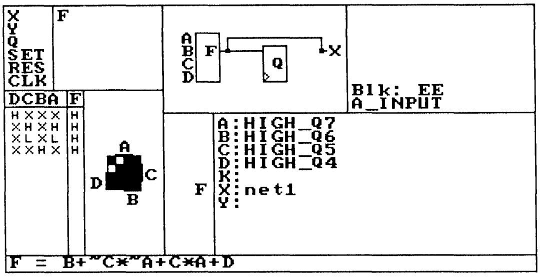


Figure 3. Shift Register Macro

4-BIT INPUT (FROM PRN)				HEXADECIMAL OUTPUT	SEGMENTS							
D	C	B	A		A	B	C	D	E	F	G	DP
0	0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	0	1	1	0	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1	0
0	0	1	1	3	1	1	1	1	0	0	1	0
0	1	0	0	4	0	1	1	0	0	1	1	0
0	1	0	1	5	1	0	1	1	0	1	1	0
0	1	1	0	6	1	0	1	1	1	1	1	0
0	1	1	1	7	1	1	1	0	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1	0
1	0	0	1	9	1	1	1	1	0	1	1	0
1	0	1	0	A	1	1	1	0	1	1	1	1
1	0	1	1	B	1	1	1	1	1	1	1	1
1	1	0	0	C	1	0	0	1	1	1	0	1
1	1	0	1	D	1	1	1	1	1	1	0	1
1	1	1	0	E	1	0	0	1	1	1	1	1
1	1	1	1	F	1	0	0	0	1	1	1	1

Table I. Decoding Table

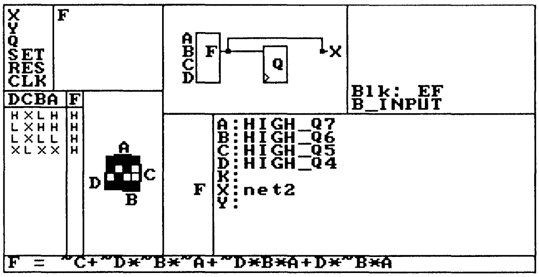


Blk: EE
A_INPUT

DCBA	F
H X X X	H
X H X H	H
X L X L	H
X X H X	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net1
Y:

F = B+~C*~A+C*A+D
```

Figure 4a. A-Segment CLB

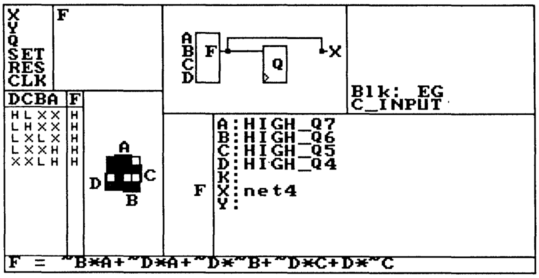


Blk: EF
B_INPUT

DCBA	F
H X L H	H
L X H H	H
L X L L	H
X L X X	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net2
Y:

F = ~C+~D*~B*~A+~D*B*A+D*~B*A
```

Figure 4b. B-Segment CLB

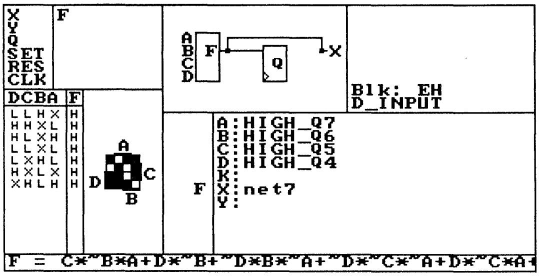


Blk: EG
C_INPUT

DCBA	F
H L X X	H
L H X X	H
L X L X	H
L X X H	H
X X L H	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net4
Y:

F = ~B*A+~D*A+~D*~B+~D*C+D*~C
```

Figure 4c. C-Segment CLB

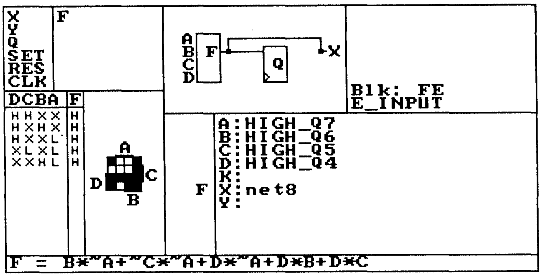


Blk: EH
D_INPUT

DCBA	F
L L H X	H
H H X L	H
H L X H	H
L L X L	H
L X H L	H
H X L X	H
X H L H	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net7
Y:

F = C*~B*A+D*~B+~D*B*~A+~D*~C*~A+D*~C*A+
```

Figure 4d. D-Segment CLB

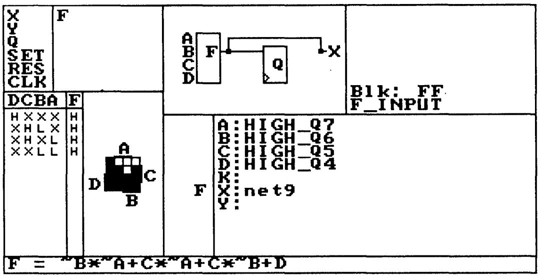


Blk: FE
E_INPUT

DCBA	F
H H X X	H
H X H X	H
H X X L	H
X L X L	H
X X H L	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net8
Y:

F = B*~A+~C*~A+D*~A+D*B+D*C
```

Figure 4e. E-Segment CLB

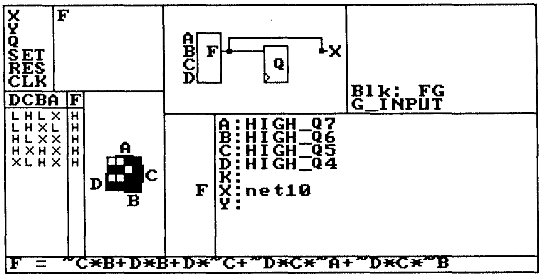


Blk: FF
F_INPUT

DCBA	F
H X X X	H
X H L X	H
X H X L	H
X X L L	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net9
Y:

F = ~B*~A+C*~A+C*~B+D
```

Figure 4f. F-Segment CLB

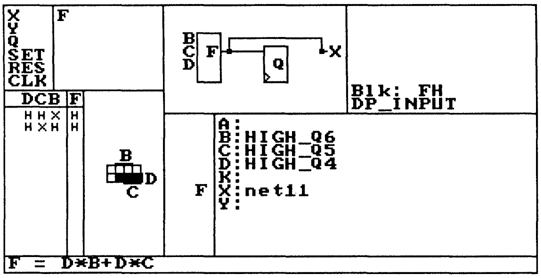


Blk: FG
G_INPUT

DCBA	F
L H L X	H
L H X L	H
H L X X	H
H X H X	H
X L H X	H

```
A:HIGH_Q7
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net10
Y:

F = ~C*B+D*B+D*~C+~D*C*~A+~D*C*~B
```

Figure 4g. G-Segment CLB



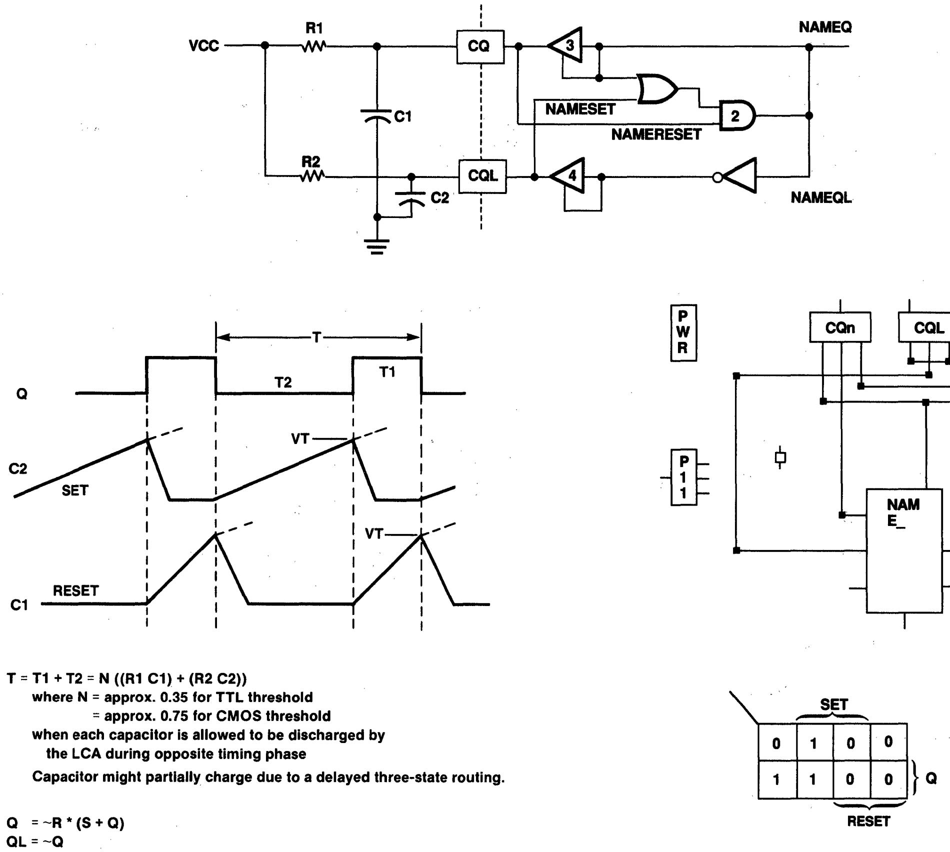
Blk: FH
DP_INPUT

DCB	F
H H X	H
H X H	H

```
A:
B:HIGH_Q6
C:HIGH_Q5
D:HIGH_Q4
K:
X:net11
Y:

F = D*B+D*C
```

Figure 4h. DP-Segment CLB

Figure 4. Seven Segment Decoder



T = T1 + T2 = N ((R1 C1) + (R2 C2))
where N = approx. 0.35 for TTL threshold
= approx. 0.75 for CMOS threshold
when each capacitor is allowed to be discharged by the LCA during opposite timing phase
Capacitor might partially charge due to a delayed three-state routing.

Q = ~R * (S + Q)
QL = ~Q

Figure 8a. Clock Macro

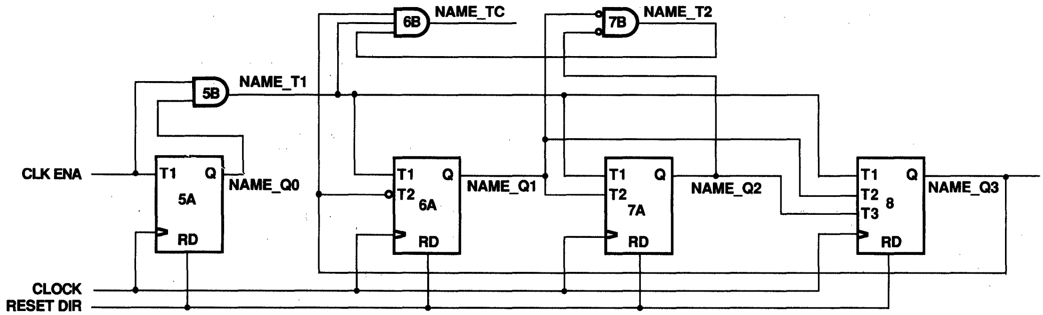
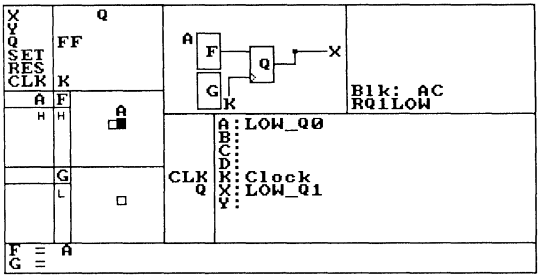


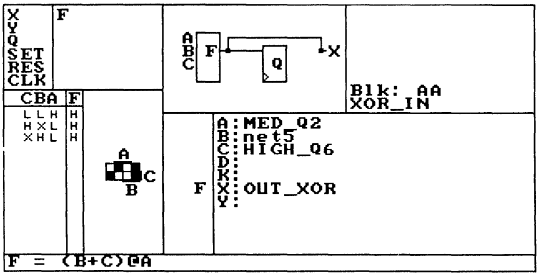
Figure 8b. BCD Counter Macro

Figure 8. Register Clock Components



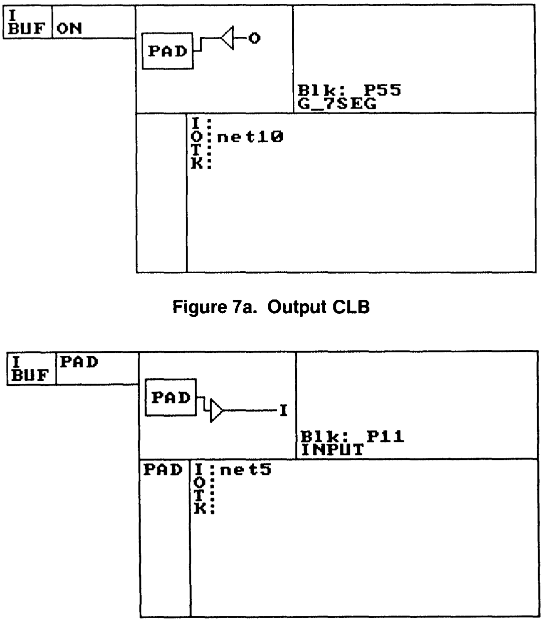
```
X        Q
Y
Q     FF
SET
RES
CLK   K
  A  F
  H  H

     G
     L
A:LOW_Q0
B:
C:
D:
K:Clock      CLK
X:LOW_Q1     Q
Y:
Blk: AC
RQ1LOW
F = A
G =
```

Figure 5. Shift Register CLB



```
X     F
Y
Q
SET
RES
CLK
 CBA  F
 LLH  H
 HXL  H
 XHL  H
A:MED_Q2
B:net5
C:HIGH_Q6
D:
K:
X:OUT_XOR    F
Y:
Blk: AA
XOR_IN
F = (B+C)@A
```

Figure 6. XOR _ F CLB



```
I
BUF   ON
I:
O:net10
T:
K:
Blk: P55
G_7SEG
```

Figure 7a. Output CLB

```
I
BUF   PAD
PAD  I:net5
     O:
     T:
     K:
Blk: P11
INPUT
```

Figure 7b. Input CLB

Figure 7. I/O CLB

PIN	NAME	DESCRIPTION
1,35	GND	Ground
52,18	VCC	Power
11	INPUT	Input seed
12,14	CQ,CQL	rc oscillator inputs
13	HA_7seg	Decoder outputs for segments in high order display (a)
15	HB_7seg	b segment
16	HC_7seg	c segment
17	HD_7seg	d segment
19	HE_7seg	e segment
21	HF_7seg	f segment
23	HG_7seg	g segment
43	HDP_7seg	dp segment
38	A_7seg	Decoder outputs for segments in low order display (a)
39	B_7seg	b segment
40	C_7seg	c segment
47	D_7seg	d segment
49	E_7seg	e segment
53	F_7seg	f segment
55	G_7seg	g segment
20	DP_7seg	dp segment
29	LSBIT_out	Least significant PRN bit
30	BIN6_out	PRN bit
31	BIN5_out	PRN bit
32	BIN4_out	PRN bit
33	BIN3_out	PRN bit
34	BIN2_out	PRN bit
36	BIN1_out	PRN bit
37	MSBIT_out	Most significant PRN bit

Table II. LCA Device Pins Used and Their Description

Conclusion

From the PRN Generator design, the advantages of using the LCA device in such an application were shown. The configurability of the LCA device makes it very versatile. The M2064 was capable of replacing several parts that normally would be needed to implement this design. The availability of the macros also helped considerably in the design implementation. It is always possible to modify certain features in this design if needed, such as the clock frequency or the size of the PRN generated, by simply re-routing certain lines in the LCA device. These features make the use of the LCA device very desirable. This design file is available from Monolithic Memories upon request. Please ask for XDES16.LCA.

References

1. Nadia Sachs, *"Pseudo-Random-Number Generator (a Disguised PAL),"* Monolithic Memories Application Note AN-118.
2. LCA device Macrocell Library Book, Monolithic Memories Inc.
3. Edward Valleau, *"Configuring the LCA Device,"* Monolithic Memories Application Note AN-182.

Advanced
Micro
Devices

AN-185

64K Deep FIFO - Dynamic RAM Controller is Implemented in the M2018 LCA Device

Karen Spesard and Chris Jay

Abstract

First-in, first-out (FIFO) buffers are used extensively in interface and communication systems where it is necessary to provide temporary storage between two asynchronously operating devices. For relatively shallow FIFOs, up to 128 locations deep, dedicated register-based FIFOs are the best solution. For medium and deep organizations, up to 64K deep, the register architecture would be unsuitable because of the high package count required for implementation. Also, with a large number of registered FIFO devices, power dissipation could be very high and lead to system reliability problems. Using the high-storage density of RAM and controlling it as a FIFO (in other words, a FIFO-RAM Controller) solves the problem of a high package count in medium to large FIFO arrays.

The two types of RAM to be considered are static RAM (SRAM) and dynamic RAM (DRAM). SRAMs usually require more board space to produce large memory arrays because their packages are larger than DRAM packages. As a result, SRAMs may be used for medium-sized FIFO organizations, from 128 bytes to 8K bytes. For large FIFO arrays, SRAM devices occupy too much board space. DRAMs, used as stand-alone memory cells require the support of more components to handle multiplexing and refresh, whereas SRAMs can be used without that support. To address this issue, a low-chip count, low-cost DRAM solution (FIFO-DRAM Controller) has been developed for large FIFO buffers in the M2018 Logic Cell™ Array (LCA device). The LCA device addresses the DRAM memory chips as a FIFO array and provides the interface and refresh control signals to the DRAMs.

The LCA device, a high-density programmable CMOS device, has been programmed to perform all of the necessary logic functions for a large FIFO - DRAM Controller. It can control 64K X 4, X 8, X 16, X 32 (and so on) DRAMs. The M2018 design handles the refresh, read and write functions as well as handshake activity to external circuits. Status flags, such as FIFO full and empty, are also provided to the two asynchronous transmitter and receiver circuits. In addition, since the LCA circuit is reconfigurable, the design can be modified as needed to meet specific design requirements.

10255A
FEBRUARY 1988

64K Deep FIFO - Dynamic RAM Controller is Implemented in the M2018 LCA Device

Karen Spesard and Chris Jay

Introduction

Data communication often requires buffering of large amounts of information between asynchronously operating devices along a data channel. For instance, in a video teleconferencing system, one Mbyte of data per video image (1K x 1K pixels) is generated. After compression, 32 Kbytes of each image must be passed to a buffer to wait for transmission along the data channel. The next image is then free to be processed by the system. For optimal performance, first-in, first-out (FIFO) buffers can be effectively used to hold the data in temporary storage sites between the transmitter and receiver nodes. The data can be stored either in a stack of registers organized as a FIFO, or a RAM controlled as a FIFO. For very large FIFO organizations, however, the use of inexpensive RAM devices is the only practical alternative. It keeps the overall system size to a minimum which will, in effect, reduce the overall system cost.

Dynamic RAM (DRAM) devices, when controlled, can be used as a temporary buffer. Though relatively inexpensive and small in size, compared with static RAM (SRAM) devices, stand-alone DRAMs require more complex interface and refresh control circuitries. As a result, more components could be required in DRAM systems. To reduce the overall chip count, a Logic Cell Array (LCA device) has been programmed to perform as a 64K x 4, x 8, x 16, x 32, (and so on) FIFO-DRAM Controller. The Controller allows the DRAM to function as a FIFO by providing the control circuitry necessary for operation.

With the design implemented in the M2018 LCA device, minor changes in the design can be implemented quickly and easily due to the device's re-programmability feature. For example, if the type of DRAM or any of its parameters had to be changed for any reason, the original LCA device can accommodate the necessary logic modification.

The LCA device provides an intelligent approach to designing circuits in an efficient and timely manner. It can speed up the design process significantly (to just a few days) with low-cost user-friendly software tools that do not require a non-recurring expense (NRE), as gate arrays often do. As a result, many products can be introduced quickly and economically. The LCA device can automatically load itself and is field programmable and reconfigurable. Therefore, it can be used for system development, for existing design modifications and for volume production. This application note describes the design methodology used for the development of the Controller in the LCA device.

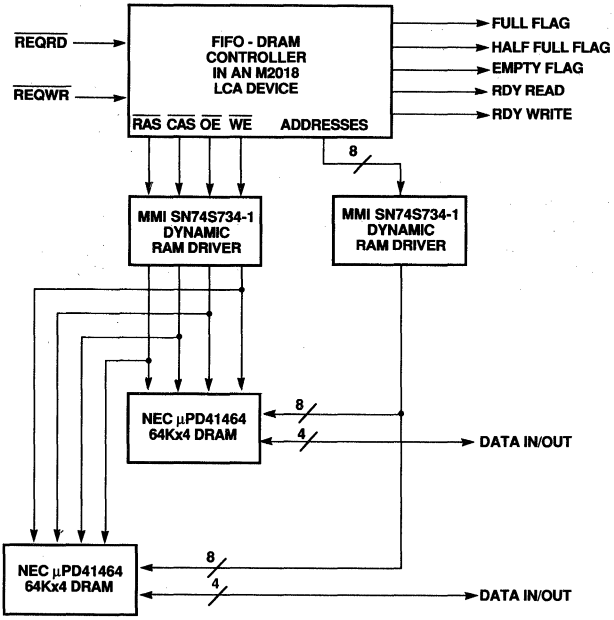


Figure 1. A Typical DRAM Interface Using the FIFO-DRAM Controller in the M2018 LCA Device

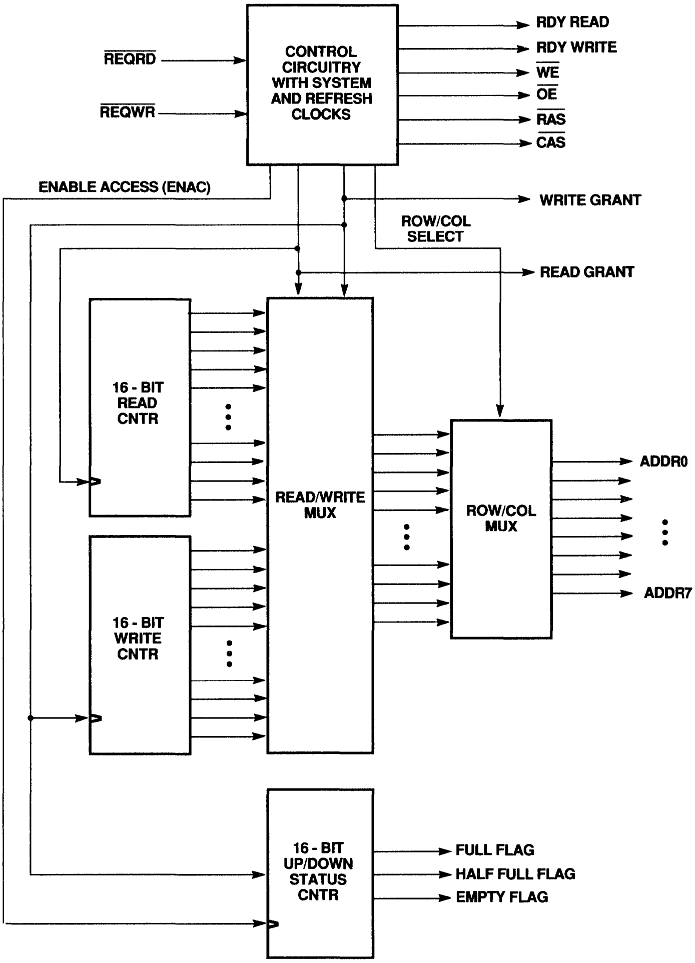


Figure 2. Block Diagram of the FIFO-DRAM Controller

The FIFO-DRAM Controller

The FIFO - DRAM Controller in the LCA device, together with an array of two DRAMs and two DRAM drivers, comprises a FIFO memory. (See Figure 1.) The Controller handles the row and column address multiplexing by applying a RAS before CAS cycle for memory read and write operations. Refresh timing, however, is applied as a CAS before RAS cycle, which takes precedence over read/write activity. The DRAMs used for this application must have on-chip refresh counters. The μPD41464 64K x 4 DRAMs manufactured by NEC were used in this application because they have a CAS before RAS internal address refresh mode. Thus, no refresh counter was needed in the design of the FIFO-DRAM Controller. The Controller in the LCA device also provides for access and refresh timing via the control signals, RAS, CAS, OE, and WE, and provides for read/write status flags.

The block diagram of the controller is shown in Figure 2. The only controller inputs are the request-to-read (REQRD) and request-to-write (REQWR) signals. These signals are active LOW and come from some external logic or a microprocessor. They drive the control circuitry section of the device, which generates the CAS and RAS signals, among others. The FIFO-DRAM Controller in the LCA device also consists of an address generation section and a buffer status section. Two internal 16-bit counters and 24 2:1 multiplexers generate the addresses for the DRAM array. A 16-it up/down counter generates the status information for the three status flags: full, empty, and half-full.

The Control Circuitry

The control circuitry provides the ready-to-read (RDY READ) and ready-to-write (RDY WRITE) signals for the system. Other signals include the row/column select (ROW/COL SELECT) signal for multiplexing the addressed data, the enable access (ENAC) signal for clocking the status counter, and the read and write grant signals for the three 16-bit counters. It also provides the four active low signals which directly interface with the DRAM: the row address strobe ($\overline{\text{RAS}}$), the column address strobe ($\overline{\text{CAS}}$), the write enable ($\overline{\text{WE}}$), and the output enable ($\overline{\text{OE}}$) signals.

The logic used for generating each of the control signals is shown in Figure 3. Much of this logic was derived from the published input signal specifications of the DRAM. For example, the $\overline{\text{RAS}}$ (A), ROW/COL SELECT (B), and $\overline{\text{CAS}}$ (C) logic was created from the write cycle, read cycle, and CAS before RAS refresh cycle timing waveforms published in the NEC μPD41464 datasheet. Samples of these timing cycles are given in Figures 4, 5, and 6. The state diagrams were then produced from these waveforms as in Figure 7A. Next, the state excitation maps or Karnaugh maps for each signal were constructed. Reducing the logic for each map and "OR"ing the enable access and enable refresh cycles together, produced the appropriate registered equations as in Figures 7B, 7C, and 7D. The logic for the $\overline{\text{OE}}$ and $\overline{\text{WE}}$ signals were generated in a similar manner.

To resolve simultaneous read, write, and refresh requests, arbitration logic was added. This was done while creating logic for the other control outputs. The logic designed in the controller gives the refresh request highest priority and the read request the lowest priority. (Refresh requests should have the highest priority because data integrity must be maintained. Write requests have the next highest priority because data is typically held on the data bus for a specific period of time and must be written before being lost.) Therefore, the read grant signal will not go HIGH until refresh and/or write cycle requests already pending have been completed. Likewise, the write grant signal will not go HIGH until a pending refresh cycle request has been completed. Also, when a refresh cycle is requested, both the RDY READ and RDY WRITE signals will be held LOW until the refresh cycle has been finished. This insures that any read and/or write requests already in queue will be completed first. The timing diagram illustrating the refresh and write priority is shown in Figure 8.

The request-to-read and request-to-write registers are reset after every read and write operation, respectively. Accordingly, the refresh register is also reset after every refresh operation. When no requests are pending, a register "sets" or holds the $\overline{\text{RAS}}$, ROW/COL SELECT, and $\overline{\text{CAS}}$ signals HIGH. The signals are now ready for the next request.

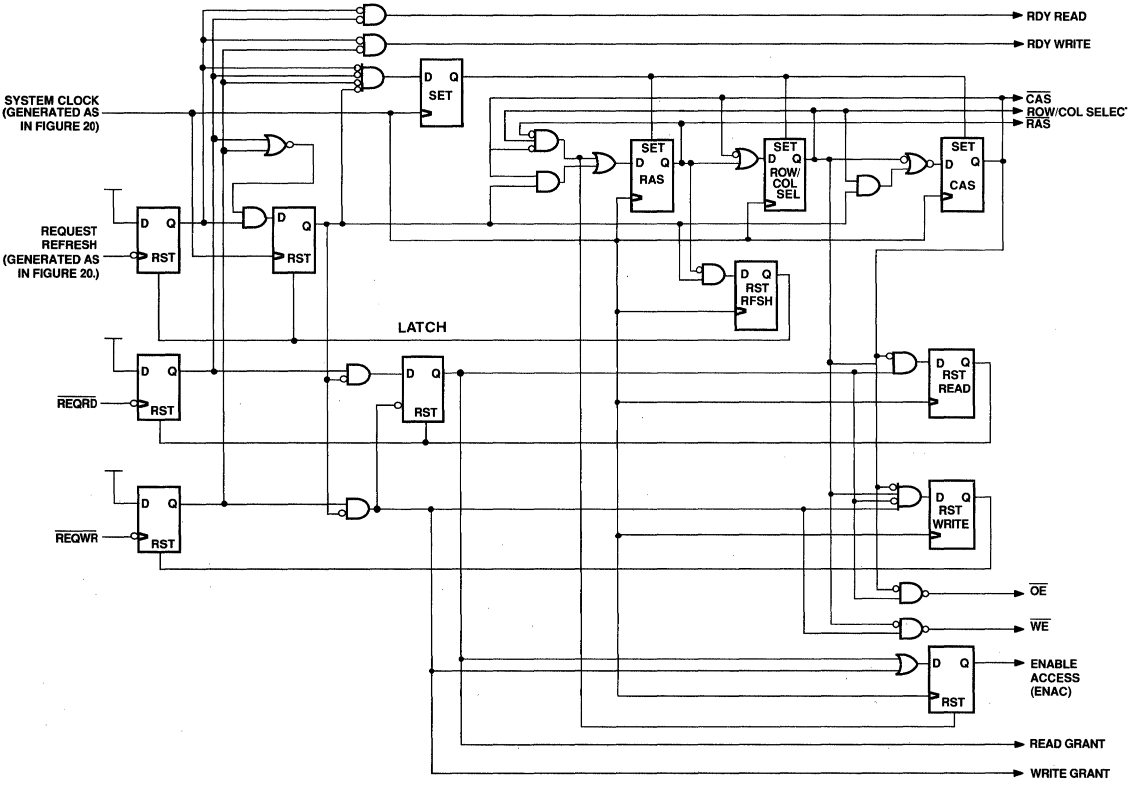


Figure 3. Control Circuitry for the FIFO-DRAM Controller

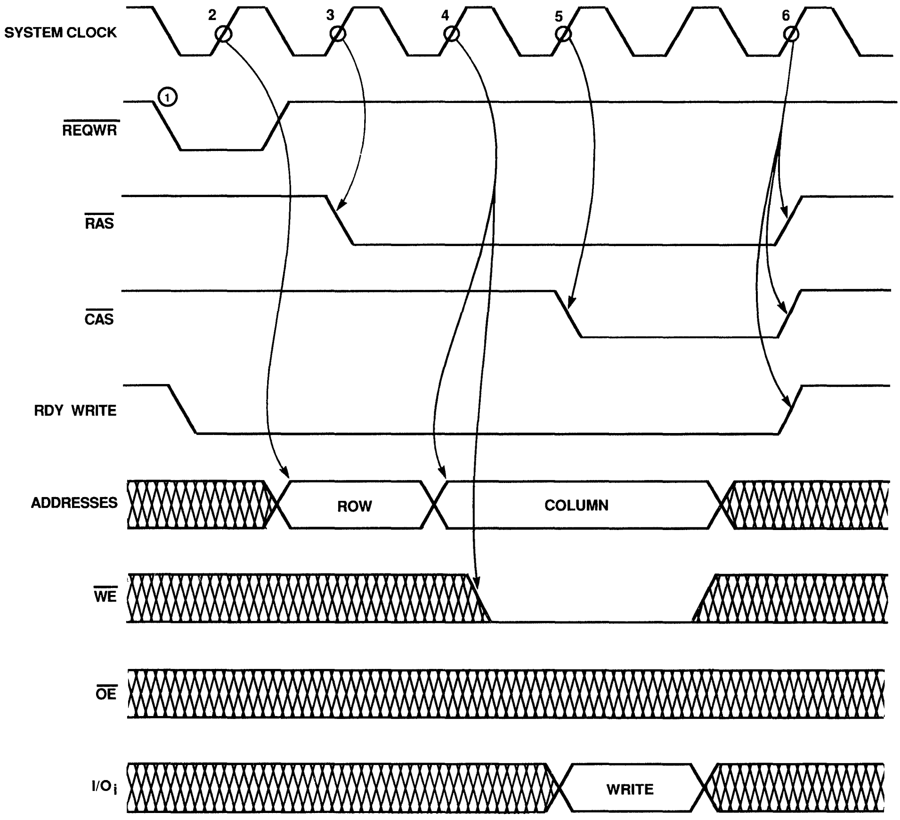


Figure 4. Write Cycle Timing

Figure 4.

1. The request to write REQWR line transitions LOW to enable an active write cycle. An active LOW transition starts the write process. The ready RDY WRITE output goes LOW to indicate that a current write cycle is in progress and no further writes should be attempted until it goes HIGH again.

2. The first active HIGH transition of the system clock sets the ROW address outputs from the write counter to the address output pins.

3. The second clock rising edge enables the active LOW transition of the RAS output to the DRAM array.

4. The third clock edge switches an internal address multiplexer which sets up a valid column address output on the input to the memory address lines. This signal is internally gated with the request to write flag to produce an early LOW active write output signal to the DRAM devices.

5. The fourth clock edge strobes the CAS output with the WE input LOW for a valid write. The OE input can be either a HIGH or LOW during this write cycle.

6. The internal state machine controlling RAS and CAS activity holds the WE LOW for one and a half clock cycles and the RAS and CAS line LOW for two clock cycles. The CAS and RAS lines are taken inactive simultaneously. The ready write signal, RDY WRITE goes HIGH to indicate that another write cycle may commence.

Note:

The frequency of the system clock is determined by the selection of external components which are configured to an internal oscillator. Details of this are shown in Figure 20.

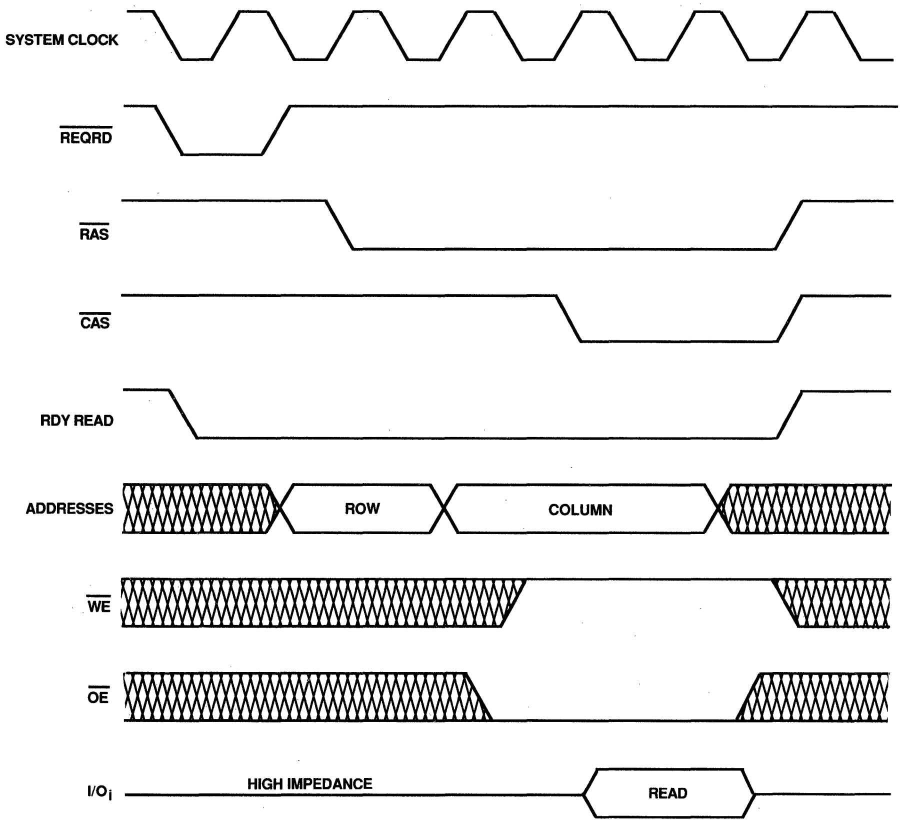


Figure 5. Read Cycle Timing

Figure 5.

1. The request to read $\overline{\text{REQRD}}$ input transitions LOW to enable an active read cycle the ready to read output, RDY READ, from the FIFO DRAM Controller goes LOW to indicate that a read cycle is in progress and no further read cycles should be requested until the read cycle is completed.

The timing of row address valid, active $\overline{\text{RAS}}$, column address valid and active $\overline{\text{CAS}}$ is identical to that shown in Figure 4. The only difference in the timing is that the $\overline{\text{WE}}$ line stays inactive HIGH and the $\overline{\text{OE}}$ signal is gated active LOW one clock period after the LOW transition of $\overline{\text{RAS}}$.

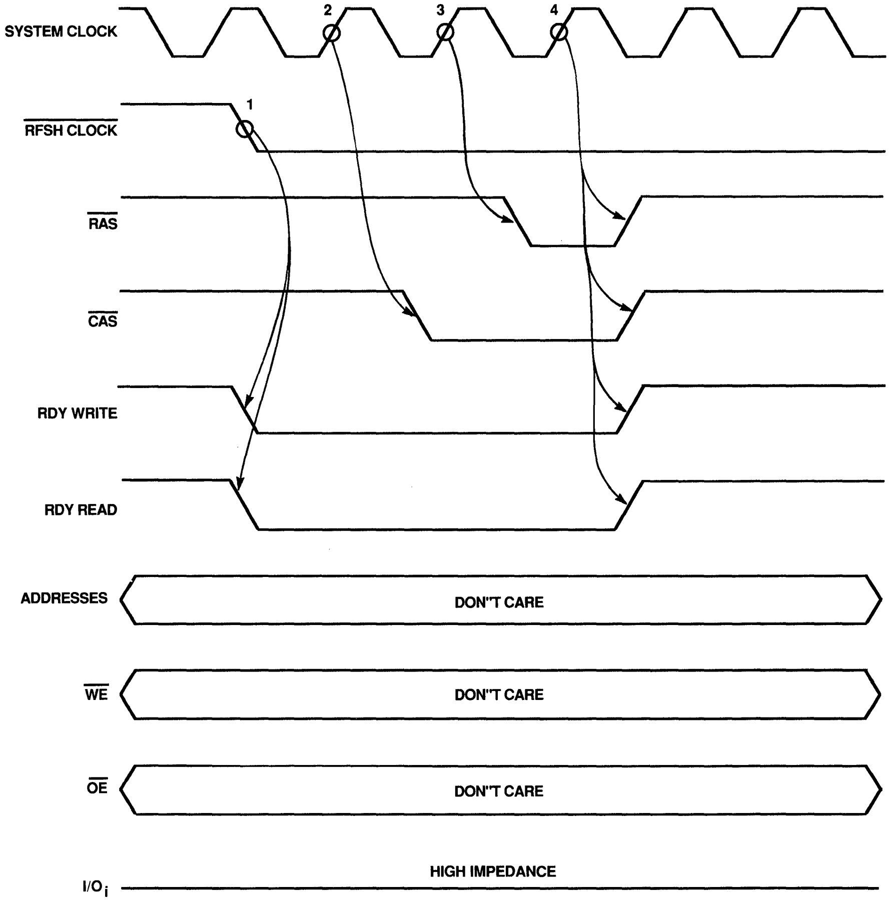


Figure 6. $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh Cycle Timing

Figure 6.

1. The $\overline{\text{RFSH}}$ CLOCK is driven active LOW to initiate a $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ refresh cycle. Ready to Read and Ready to Write Controller outputs are driven LOW to allow an uninterrupted DRAM refresh.
2. The $\overline{\text{CAS}}$ output goes LOW after the first clock edge.
3. The $\overline{\text{RAS}}$ output goes LOW following the second clock edge.
4. The $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ outputs go inactive after the third clock edge and RDY WRITE and RDY READ go HIGH to indicate that read and write operations may recommence.

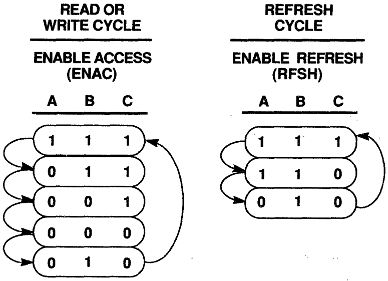
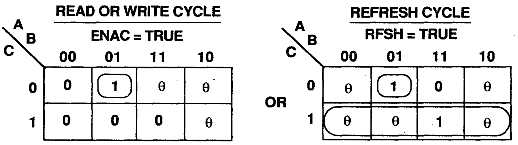


Figure 7a. State Diagrams for Read and Write Access Enable (ENAC) and Refresh Enable (RFSH) where A = RAS Signal, B = ROW/COLUMN ADDRESS SELECT Signal, and C = CAS Signal



READ OR WRITE CYCLE, ENAC = TRUE

C \ AB	00	01	11	10
0	0	1	θ	θ
1	0	0	0	θ

OR

REFRESH CYCLE, RFSH = TRUE

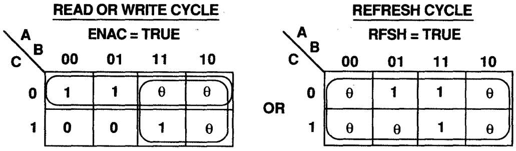
C \ AB	00	01	11	10
0	θ	1	0	θ
1	θ	θ	1	θ

A := ~A * B * ~C (ENAC + RFSH) + (RFSH * C

A := ~A * B * ~C + RFSH * C

RAS := ~RAS * ROW/COL SEL * ~CAS + RFSH * CAS

Figure 7b. State Excitation Map for RAS (A) Signal and Corresponding Equation



READ OR WRITE CYCLE, ENAC = TRUE

C \ AB	00	01	11	10
0	1	1	θ	θ
1	0	0	1	θ

OR

REFRESH CYCLE, RFSH = TRUE

C \ AB	00	01	11	10
0	θ	1	1	θ
1	θ	θ	1	θ

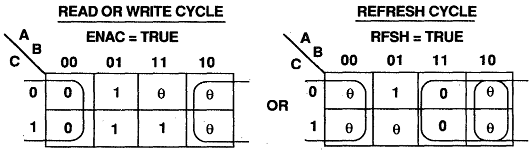
B := ~C * ENAC + A * ENAC + RFSH

B := ~C * A (ENAC + RFSH)

B := ~C * A

ROW/COL SELECT := ~CAS * RAS

Figure 7c. State Excitation Map for ROW/COLUMN ADDRESS SELECT (B) Signal and Corresponding Equation



READ OR WRITE CYCLE, ENAC = TRUE

C \ AB	00	01	11	10
0	0	1	θ	θ
1	0	1	1	θ

OR

REFRESH CYCLE, RFSH = TRUE

C \ AB	00	01	11	10
0	θ	1	0	θ
1	θ	θ	0	θ

C := ~ (~B* ENAC) + ~ (A * RFSH)

C := ~ (~B * (ENAC * RFSH) + ~ (A* RFSH)

C := ~ (~B + A * RFSH)

CAS := ~ (~ROW/COL SELECT + RAS * RFSH)

Figure 7d. State Excitation Map for CAS (C) Signal and Corresponding Equation

Figure 7.

1. Figure 7a shows the truth tables for the controlling state machine within the FIFO DRAM controller. Two tables should be considered, one for access, read and write activity, and one for DRAM refreshing. The Enable access ENAC and enable refresh RFSH are mutually exclusive events such that ~RFSH may be considered as ENAC and ~ENAC as RFSH. When ENAC is TRUE (1) an access cycle is taking place, and RFSH will be FALSE (0). Refreshing takes place when RFSH is TRUE and ENAC is FALSE. The Karnaugh map shown in Figure 7b represents the state excitation map for the RAS output during access and refresh cycles. A logic one entry represents an active condition and a logic zero a passive state. The Greek letter theta represents a "don't care" state and can be included in minimization considerations. For the map shown in Figure 7b, minimization yields;

ENAC = 1, A := ~A*B*~C.............................. 1

RFSH = 1, A := ~A*B*~C + C 2

now, combining 1 and 2;

A := ~A*B*~C*ENAC + ~A*B*~C*RFSH + RFSH*C......... 3

reduces to

A := ~A*B*~C + RFSH*C.............................. 4

because

ENAC + RFSH = 1.

substituting the values of RAS, ROW/COL and CAS for A, B and C respectively we get;

RAS := ~RAS*(ROW/COL SEL)*~CAS + RFSH*CAS

The equations for ROW/COL SEL and CAS are derived in a similar way.

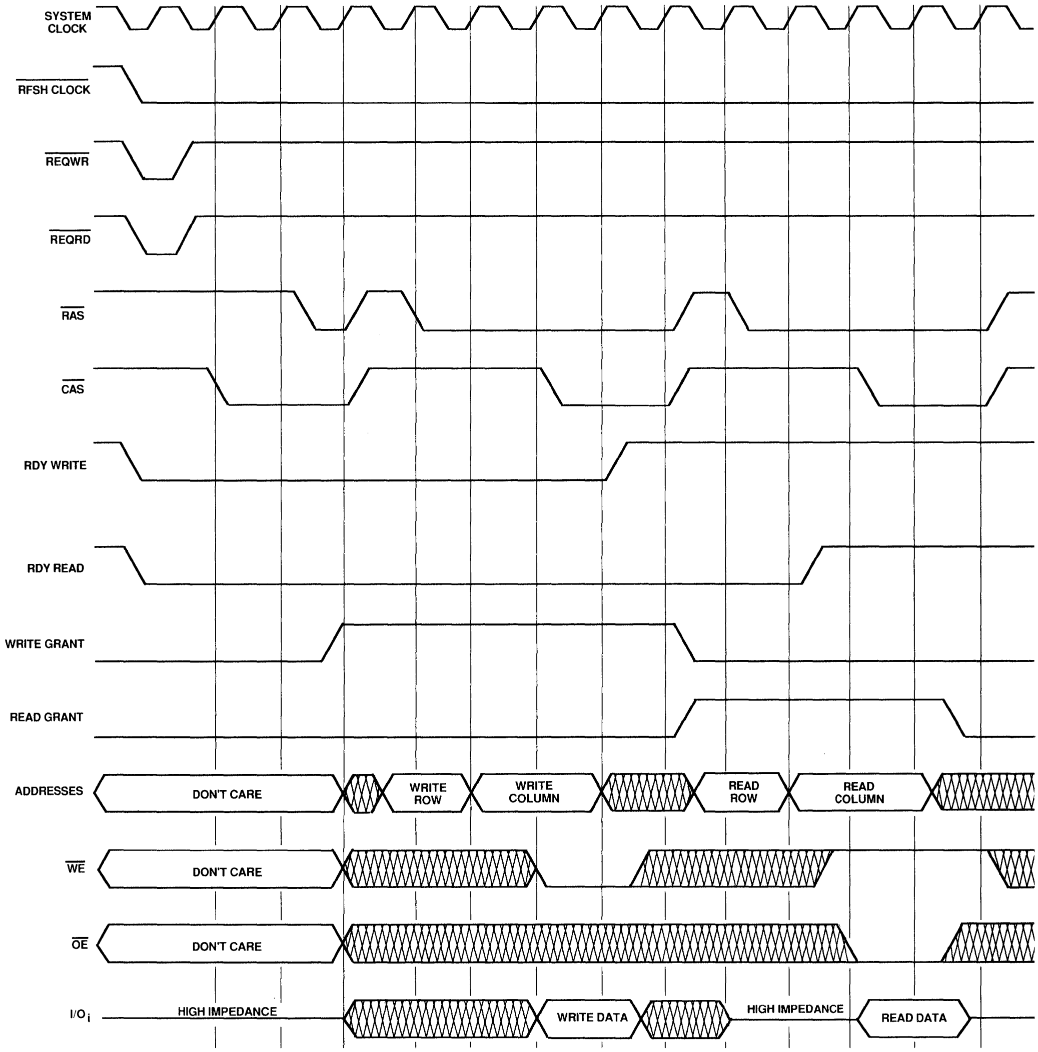


Figure 8. Timing with Refresh and Write Priority

The system and refresh clocks are generated internally with two resistor-capacitor oscillator networks each. The system clock can be designed to run as fast as 5 MHz. The refresh clock, which is the request to refresh signal, is designed to operate at 15.6 μs or 64 KHz. This is because the DRAM must be refreshed with 256 refresh cycles in a period of 4 ms. (The implementation of the oscillator network will be described in the System Clock and Refresh Request Clock Configurations section.)

READ/WRITE Address Circuitry

The FIFO-DRAM Controller in the LCA device is based on a FIFO architecture and consists of a read and write pointer. The addresses of these pointers are generated by two 16-bit counters and 24 2:1 multiplexers, as shown in Figure 9. When the write grant signal is activated, the first 16-bit counter is incremented. The write address is then multiplexed through to the row/column select multiplexer. As the ROW/COL SELECT signal goes HIGH, the first eight bits of the counter (the row address of the DRAM array) are transferred onto the address bus, and as the ROW/COL SELECT signal goes LOW, the second eight bits of the counter (the column address of the DRAM array) are transferred onto the address bus. An equivalent sequence also occurs when a read grant signal is activated.

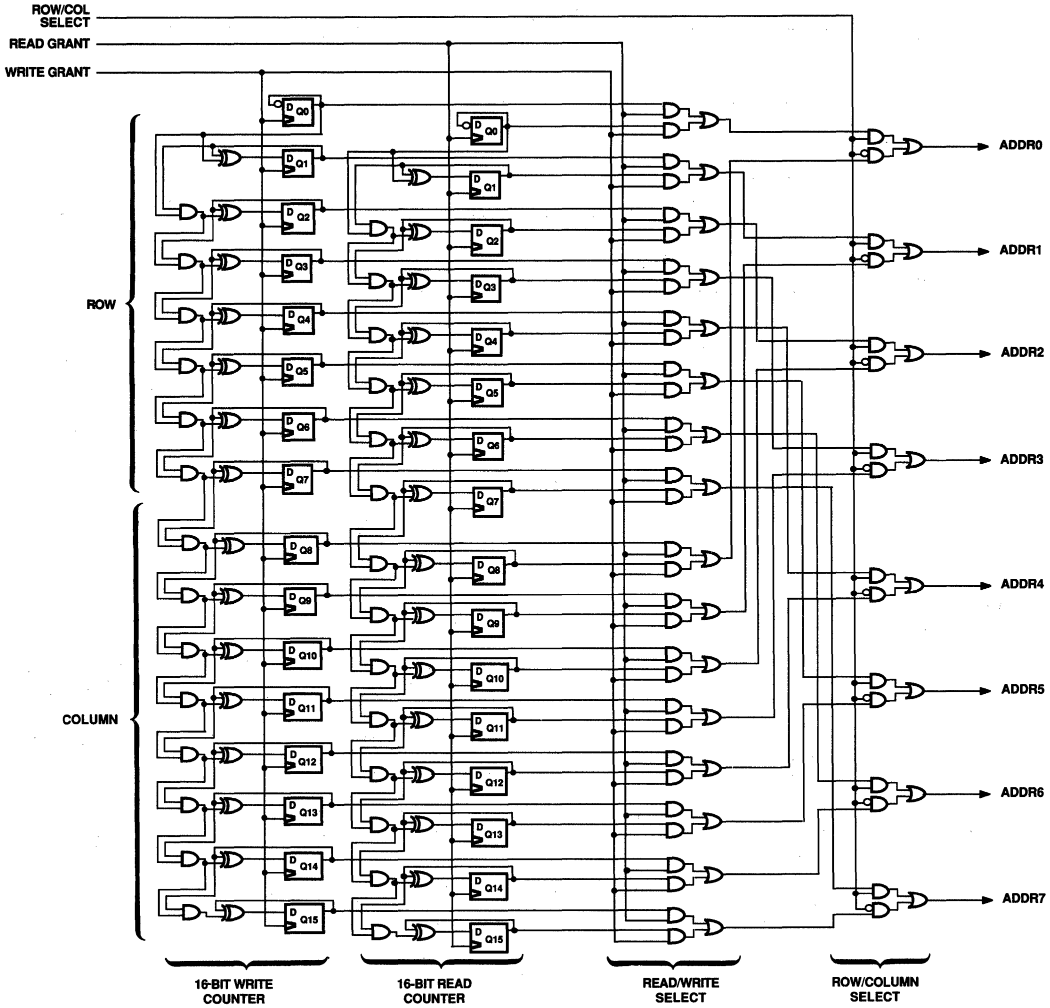


Figure 9. Read/Write-Row/Column Address Generation

The Status Counter

Information about the FIFO buffer is provided by the three status flags: full, half-full, and empty, which are all active HIGH. The full and empty flags indicate overflow and underflow conditions, whereas the half-full flag gives an indication to a processor of the speed at which the buffer is being filled. It acts as a monitor rather than as a prevention flag.

The flags are constructed from the status counter, a 16-bit up/down counter shown in Figure 10. The clock input to the counter is the enable access signal that "OR"s the write grant and read grant signals. When the write grant signal is HIGH, the counter is incremented, and when the write grant signal is LOW, the counter is decremented. For example, if the count reaches 0000 Hex during a read operation, the FIFO buffer is empty, and if the count reaches FFFF Hex during a write operation, the buffer is full. If the count reaches 7FFF Hex, the buffer is half-full.

Logic Cell Array Implementation

The FIFO-DRAM Controller design was implemented in one chip: the M2018 Logic Cell Array, a high-density, programmable CMOS device. This device contains 100 Configurable Logic Blocks (CLBs) containing one register element each. The register element can be configured as a D-type flip-flop or latch, or it can be bypassed totally, supplying a strictly combinational output. The two available CLB outputs can be configured as a function of four Boolean input variables, or two functions of three input variables each.

The M2018 also contains 74 configurable input/output blocks (IOBs). Sixty-four of these blocks can be configured to perform a variety of logic functions. Each has the capability to drive an output, receive an input, clock the input into a flip-flop, or provide both input and output capability under three-state control. Together, the CLBs and the IOBs provide 1800 usable logic gates. These gates are configured when data is loaded into the LCA device for programming — usually from an EPROM or microprocessor. For example, the configuration data file, <filename>.prm, to be programmed into an EPROM, is generated using the program, MAKEPROM (part of Monolithic Memories' XACT™ Development System), which loads the bit stream into PROM memory locations. The input to the MAKEPROM program, <filename>.bit, is generated from the MAKEBITS program which outputs the bit stream of the current design. The design is entered by either using one of the schematic capture-based systems from Daisy, Mentor Graphics®, OrCAD™, and Futurenet® or by partitioning the required logic and inserting it in a graphics environment on the IBM® PC-XT/AT® using Monolithic Memories' XACT™ LCA Editor. The LCA Editor was used to create the LCA design of the FIFO-DRAM Controller. The design implementation using CLBs and IOBs is described below.

CLB Implementation

The $\overline{\text{RAS}}$, ROW/COL SELECT, and $\overline{\text{CAS}}$ outputs are implemented in four CLBs. The RAS output utilizes two CLBs but also includes the enable access signal (ENAC2), which is reset with part of a RAS function. Figure 11 shows the CLBs that comprise the RAS output. The COMRASB signal, in Figure 11A, is configured from a three-input combinational function (the Y output-G base) and is an input to the RAS block, in Figure 11B. This signal is "OR"ed with two more signals. The RAS block, configured as a D-type flip-flop with a "set" function, produces the $\overline{\text{RAS}}$ output. The CLB configurations for the ROW/COL SELECT and $\overline{\text{CAS}}$ outputs are shown in Figures 12 and 13. The CLB equations correspond to the equations for each signal given in Figure 7.

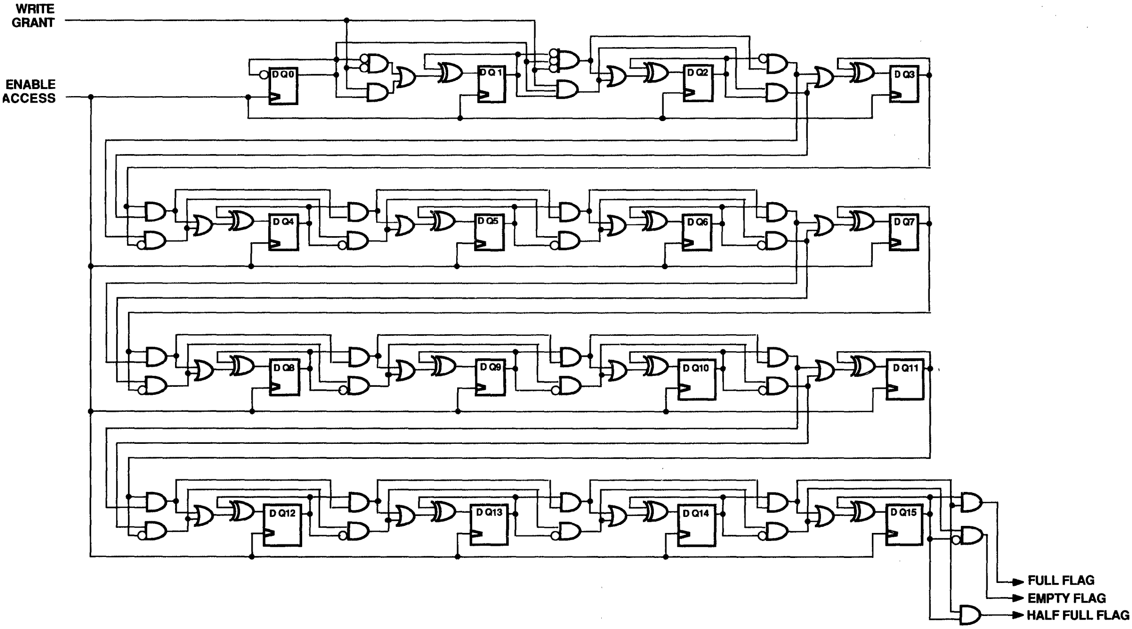
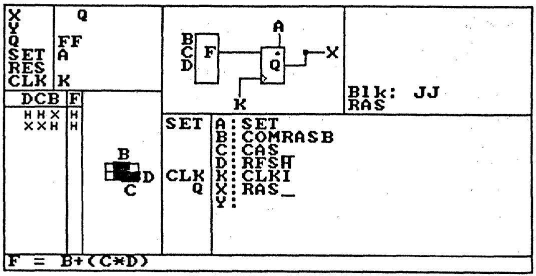
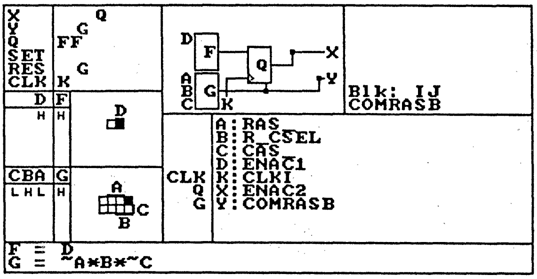


Figure 10. 16-Bit Up/Down Status Counter Circuitry



```
Blk: JJ
RAS
SET  A:SET
     B:COMRASB
     C:CAS
     D:RFSH
CLK  K:CLKI
  Q  X:RAS_
     Y:
F = B+(C*D)
```

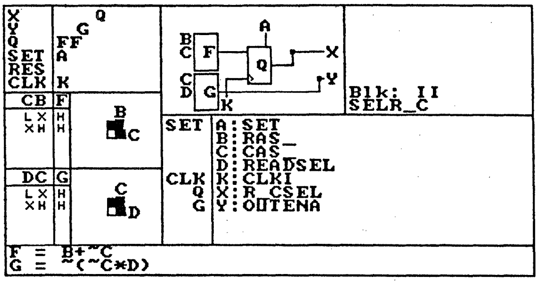
Figure 11a.



```
Blk: IJ
COMRASB
     A:RAS
     B:R_CSEL
     C:CAS
     D:ENAC1
CLK  K:CLKI
  Q  X:ENAC2
  G  Y:COMRASB
F = D
G = ~A*B*~C
```

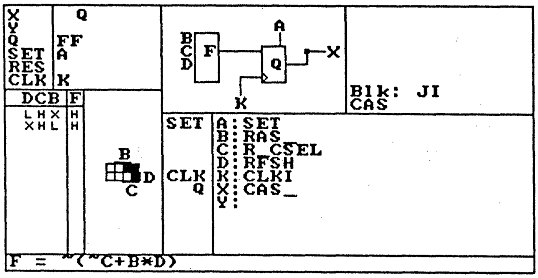
Figure 11b.

Figure 11. CLBs which Comprise the $\overline{\text{RAS}}$ Signal. The Y Output of Figure 11a, COMRASB, is the B Input to Figure 11b.



```
Blk: II
SELR_C
SET  A:SET
     B:RAS_
     C:CAS
     D:READSEL
CLK  K:CLKI
  Q  X:R_CSEL
  G  Y:OUTENA
F = B+~C
G = ~(~C*D)
```

Figure 12. CLB Configuration for the Row/Column Select Signal



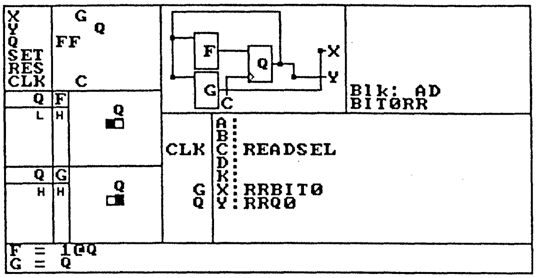
```
Blk: JI
CAS
SET  A:SET
     B:RAS
     C:R_CSEL
     D:RFSH
CLK  K:CLKI
  Q  X:CAS_
     Y:
F = ~(~C+B*D)
```

Figure 13. CLB Configuration for the $\overline{\text{CAS}}$ Signal

The CLB configuration for the 16-bit read counter is shown in Figure 14. The first (row - read) bit, RRQ0 on output Y, is illustrated in Figure 14A. It is generated through a flip-flop and is clocked with the read select signal. The terminal count for the next bit is generated from RRBIT0, the X output. Figure 14B shows the configuration for the second bit of the read counter.

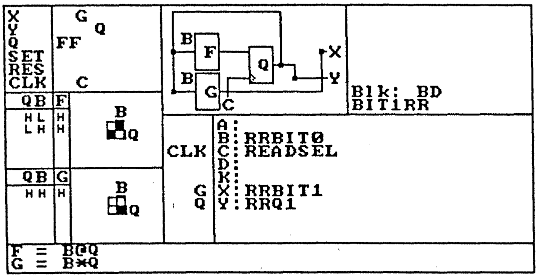
The read/write address multiplexer is designed with four Boolean input variables as in Figure 15. The RRQ0 signal (row-read) is selected if the read select signal is active HIGH, and the RWQ0 signal (row-write) is selected if the write select signal is active HIGH. This implementation is combinational and has only one output.

The row/column address multiplexer is designed with a CLB internal multiplexer base, FGM, as in Figure 16. When the ROW/-COL SELECT signal is HIGH, the ROW address, MA0, is selected, and when the ROW/COL SELECT signal is LOW, the COLUMN address, MB0, is selected. This last multiplexer is registered and outputs the address that is output to the DRAM.



```
Blk: AD
BIT0RR
     A:
     B:
CLK  C:READSEL
     D:
     K:
  G  X:RRBIT0
  Q  Y:RRQ0
F = 1@Q
G = Q
```

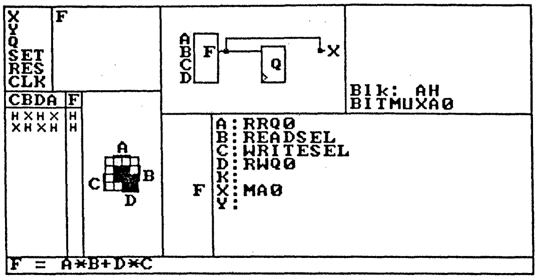
Figure 14a.



```
Blk: BD
BIT1RR
     A:
     B:RRBIT0
CLK  C:READSEL
     D:
     K:
  G  X:RRBIT1
  Q  Y:RRQ1
F = B@Q
G = B*Q
```

Figure 14b.

Figure 14. Two CLBs which Produce Bit0 and Bit1 of the Row/Read Address Bit Respectively



```
Blk: AH
BITMUXA0
     A:RRQ0
     B:READSEL
     C:WRITESEL
     D:RWQ0
     K:
  F  X:MA0
     Y:
F = A*B+D*C
```

Figure 15. CLB Configuration of the Read/Write Select Multiplexer

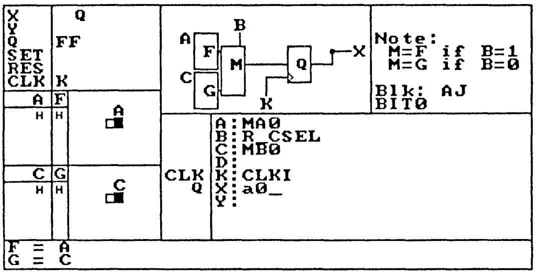


Figure 16. CLB Configuration of the Row/Column Select Multiplexer

The up/down status counter implementation is shown in Figure 17A. The second status counter bit is produced from the Y output, a registered function of input C2X, C2Y and the feedback Q. The flip-flop clock is ENAC, the DRAM access signal. The X output is a combinational function of C2X and Q. The third bit is produced from a registered equation as shown in Figure 17B.

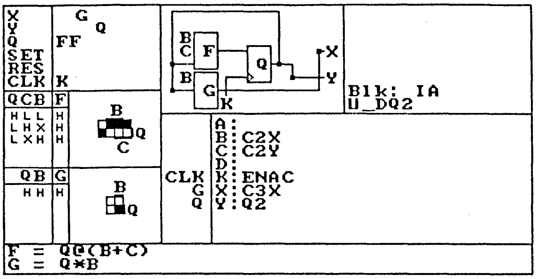


Figure 17a. The Third Bit of the Status Counter

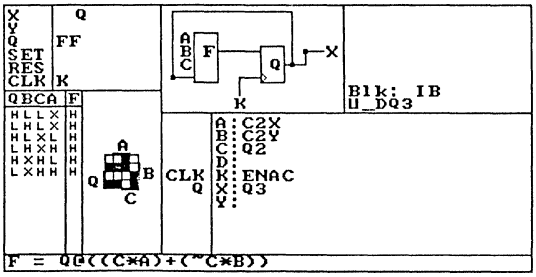


Figure 17b. The Fourth Bit of the Status Counter

Figure 17. Two CLBs which Produce the Third and Fourth Bits of the Status Counter

IOB Implementation

An example of an IOB configured to receive data and drive inputs is shown in Figure 18A for the write request (WRR) block. An IOB configured as an output buffer for the RAS signal is shown in Figure 18B. The placement of the IOBs, like the CLBs, was optimized for the best circuit layout.

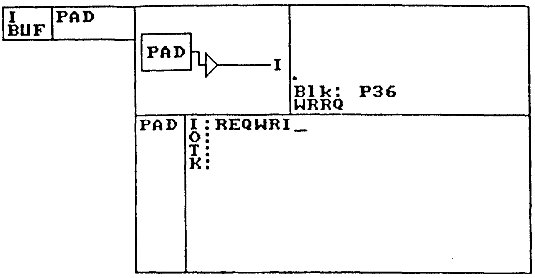


Figure 18a. The Input IOB Configuration Used for the Write Request Signal

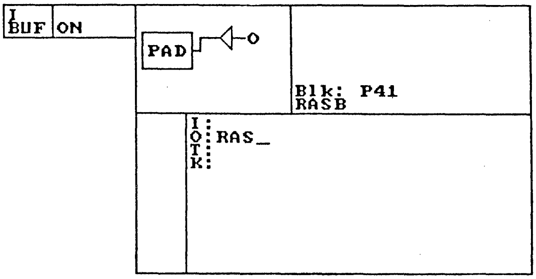


Figure 18b. The Output IOB Configuration Used for the RAS Signal

Figure 18. Two IOB Configurations Used in the FIFO-RAM Controller Design

System Clock and Refresh Request Clock Configurations

The system and refresh request clocks were designed with two general purpose oscillator networks. Each oscillator was built using two IOBs, one CLB, and two external resistor/capacitor networks — R1, C1, and R2, C2. The IOBs and CLBs were configured by calling up the dedicated macro, GOSC, from Monolithic Memories' Macrocell Library, available as part of the schematic capture entry package or the XACT EditLCA entry package. The LCA implementation of the relaxation oscillator is shown in Figure 19. Here, the programmable routing connections of CLB block RFSCLK and IOB blocks CQLrfs and CQrfs are shown for the refresh clock. Figure 20 illustrates the logic schematic for both the system and refresh clocks and the correct connections to the resistor/capacitor network.

The CLB and IOB configurations for the generation of the refresh clock are shown in Figure 21. Notice that in both the IOB configurations, the bidirectional three-state buffer is used. Notice also that in the CLB configuration, the output X, RFSQ, is the same as the input D. This is possible even though the function is not registered.

The calculation of the resistor/capacitor values for the clocks is given below. For an even mark/space ratio, R = R1 = R2 and C = C1 = C2. Each timing phase (1/2 of a cycle period) is given by the following formula:

T1 = 0.35 (C1 * R1 * 2) for TTL voltage thresholds, and

T2 = 0.75 (C2 * R2 * 2) for CMOS voltage thresholds.

The general expression for the calculation becomes:

T = N * ((R1 * C1) + (R2 * C2))

where N is .35 for TTL and 0.75 for CMOS.

The resistor and capacitor values chosen for the system clock are R1 = R2 = 30 ohms and C1 = C2 = .01 microfarads. These components are connected to pins 11 and 13 and result in a frequency of 5 Mhz. The values chosen for the refresh request clock are R1 = R2 = 2.2 Kohms and C1 = C2 = .01 microfarad. They are connected to pins 24 and 28 and result in a frequency of approximately 64 KHz. The GOSC macro IOBs can be connected to any pins as long as they are not connected to pins dedicated to configuring the LCA device. Dedicated configuration pins cannot be loaded with components such as capacitors because they could prevent the LCA device from reading valid data by corrupting hold and setup times.

Design Considerations

The circuit diagram of the FIFO-DRAM Controller in the LCA device is shown in Figure 22. This implementation used all 100 CLBs available in the M2018 device. The three columns of CLBs, on the left, contain the status counter. The next seven columns of CLBs contain the address counters and the multiplexers, and the two bottom rows of CLBs contain the control circuitry.

The placement of the blocks are positioned so that routing delays between the blocks can be minimized. Long line interconnects are used frequently for signals which would have to travel the length or breadth of the chip. These lines were also used for signals that must have minimal skew between destination points. For more information on routing, please refer to Chapter 9 of the LCA Design and Applications Handbook.

Summary

For DRAM designs, a FIFO-DRAM Controller was efficiently implemented in a single CMOS M2018 LCA device. The controller, which enables the DRAM to function as a FIFO, allows large amounts of data to be held in temporary storage. The design can be modified quickly and easily with the LCA device because it is configurable and reprogrammable. Configuration of the LCA device is discussed in the LCA Applications Note 182.

The FIFO-DRAM Controller design is available upon request. The design file and bit file will be provided for configuring the LCA device. Please ask for design XDES10.LCA.

References

NEC Electronics Inc. μPD41464 65,536 x 4-Bit Dynamic NMOS Ram Datasheet, January 1987

Monolithic Memories, Inc., FIFO-RAM Controller 57/674219 Datasheet, September 1986

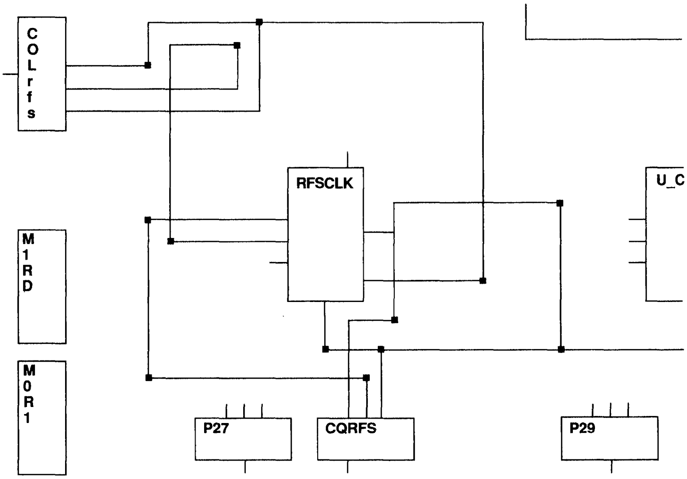


Figure 19. LCA Implementation of the Relaxation Oscilaltor with the CLB/IOB View of the External Routing Connections

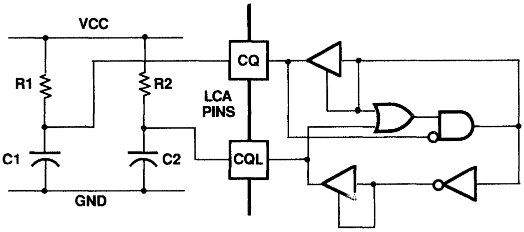
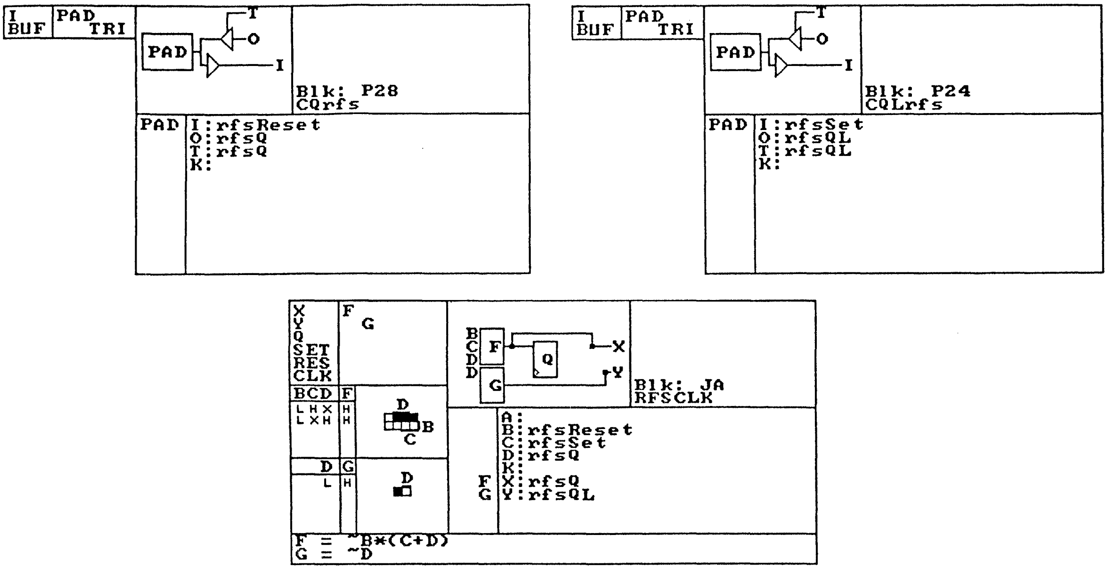


Figure 20. GOSC Macro Logic Schematic for System and Refresh Clocks



```
I    PAD
BUF     TRI
PAD  T O I
Blk: P28
CQrfs
PAD I:rfsReset
    O:rfsQ
    T:rfsQ
    K:
```

```
I    PAD
BUF     TRI
PAD  T O I
Blk: P24
CQLrfs
PAD I:rfsSet
    O:rfsQL
    T:rfsQL
    K:
```

```
X   F
Y     G
Q
SET
RES
CLK
BCD F
L H X H
L X H H
D G
  L H
B
C  F   Q   X
D
D  G       Y
Blk: JA
RFSCLK
  A:
  B:rfsReset
  C:rfsSet
  D:rfsQ
  K:
F X:rfsQ
G Y:rfsQL
F = ~B*(C+D)
G = ~D
```

Figure 21. CLB and IOB Configurations for the Refresh Clock

Print World: DRAMFIFO.LCA (2018NL68-70), XACT 1.30, 01:59:23 JAN 1, 1980Print World: DRAMFIFO.LCA (2018NL68-7

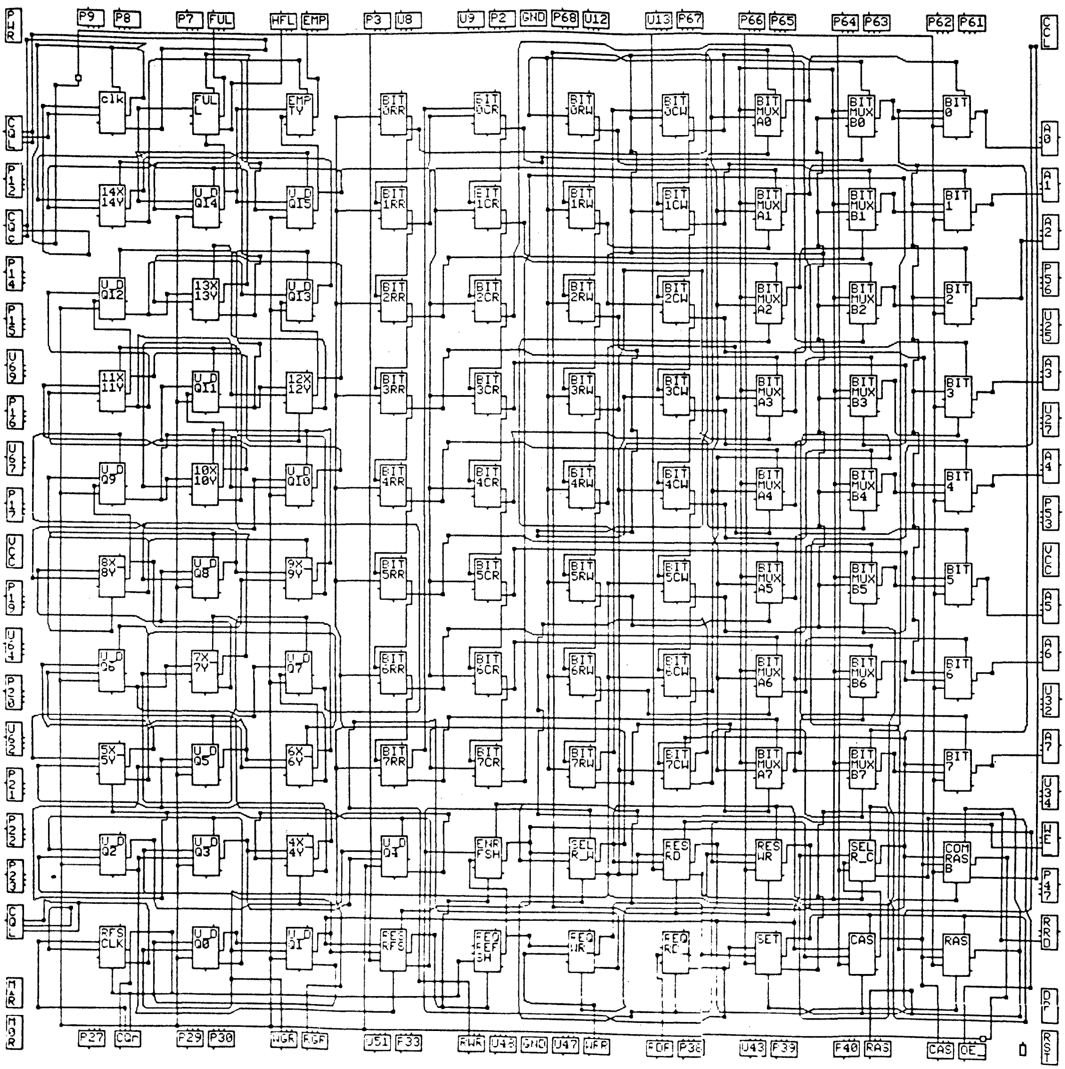


Figure 22. FIFO-DRAM Controller Circuit Implementation in the LCA Device

Advanced
Micro
Devices

Configuring the LCA™ from the PC Bus

By Robert Botchek of Trantor Systems Ltd. and Chris Jay of AMD

ABSTRACT

The Logic Cell™ Array (LCA) is a high density programmable device based on a complex arrangement of static RAM cells. The device must go through an initialization and configuration cycle each time power is applied. This cycle must occur before the LCA can function as a logic unit or subsystem. The configuration time is small, ranging from 17 to 35 milliseconds for the M2000 series of devices, and when the device is fully functional no re-configuration is necessary provided power to the device is maintained.

There are a number of modes of configuration, each suited to a particular device application. This application note gives detailed information on configuring the LCA device from the IBM PC® bus, choosing one of the five configuration modes available. Configuring the LCA device directly from the PC bus presents numerous advantages to the PC adapter card designer. The direct link between the PC bus and the LCA enhances debugging and testing, and reduces the product's time to market. The concept of soft programmability (based on the LCA's SRAM design) significantly reduces the development cycle of a PC product using the LCA. Once complete, the configuration software can exist in ROM and be called each time the system boots, so PC cards containing LCA devices can be configured in a manner transparent to the user. The interface requires both software and hardware considerations. A high level language program is written to perform data code conversion, and assembly language programs supervise the actual LCA downloading and programming cycle. The hardware interface to the LCA is achieved with a low-cost PAL16L8 device. A description of hardware and software design considerations is given in the following application note.

Publication #	Rev.	Amendment
10719	A	/0
Issue Date: May 1988		

Configuring the LCA from the PC Bus

By Robert Botchek of Trantor Systems LTD. and Chris Jay of AMD

INTRODUCTION

The Logic Cell Array (LCA) is a very high density programmable device. The 2000 series consists of two devices, the M2064 and the M2018. The M2064 consists of an eight-by-eight matrix of programmable Configurable Logic Blocks (CLBs) with fifty-eight programmmable Input/Output Blocks (IOBs) to handle a variety of input/output functions. The M2018 has a ten-by-ten matrix of CLBs with seventy-four IOBs. Both devices have a complex array of programmable interconnect which is used to connect the logic blocks and input/output functions. The M3000 series was developed from the 2000 series and is suitable for even more dense logic architechtures. The 3020 device is a development of the 2064 having an eight-by-eight array of more complex logic blocks. The IOBs and interconnect are also more complex in this new generation of programmable gate array circuits. This application note applies equally well to the configuration of 3000 series and 2000 series devices.

These high density devices require good software support tools to assist the designer in achieving a successful design with rapid turn-around, a benefit well known to the engineer familiar with Programmable Logic Devices; PLDs. Most manufacturers of PLDs support their products with at least one software package, and as complexity of programmable logic increases there is a need for more sophisticated computer-based tools. The LCA device is probably the most complex PLD on the market today, but fortunately there exists a wide repertoire of software support packages to provide the logic systems designer with rapid design entry and logic verification. In addition, conversion software provides a bridge from the finished design into raw configuration data which in turn is suitable for downloading and configuring the device. LCA device design entry is well supported in the PC environment with a wide range of CAE tools. The XACT™ software is designed to run on a PC XT/AT, for the 2000 series of LCA products only, and PC/AT for both 2000 and 3000 family of devices. Many packages for schematic entry and simulation also exist as software support for the LCA product, and will run in the PC environment.

The route from design entry to a configuration bit stream is a relatively fast process. Thus, even if a number of reprogramming cycles are needed to prove a design, the overall process remains comparatively short.

CONFIGURATION MODES

There are five LCA configuration modes. These are listed in Table 1 along with corresponding LCA mode select inputs M0-M2. For the specific application of configuring the LCA from the PC bus the slave mode is used. In this mode, for a single LCA device, only three pins take part in the configuration process. This can be the deciding factor in using the slave mode because some of the IOBs have a dual function of address and data assignment in the master low and high modes. During configuration these IOBs might need isolation from external logic. IOB intensive applications would favor the slave mode to avoid the additional time and space penalty of designing in additional logic buffers to perform this isolation.

Of the three pins used, two are required for handshake and one for data transmission. The data is set up at the DIN input to the LCA and clocked into the device by applying a rising clock edge to the CCLK input. Each configuration bit is synchronously loaded in this manner at a rate determined by the PC interface. The remaining handshake pin is the Done/~Program pin, when LOW the LCA is in configuration mode and when HIGH the configuration process has been completed and the LCA device is functionally operational.

The slave mode of configuration can also be used effectively for chaining multiple LCA designs. The DOUT pin from a preceding LCA device can be fed directly to the DIN pin of a succeeding LCA so two or more devices can be configured. Waveforms in Figure 3 show the timing requirements for DIN, DOUT and CCLK.

M2018 CONFIGURATION FILE

```
1111                                  FOUR DUMMY BITS MINIMUM
0010                                  PREAMBLE CODE
<24 BIT LENGTH COUNT>                 TOTAL NUMBER OF BITS
1111                                  FOUR DUMMY BITS MINIMUM

0<DATA FRAME NUMBER 001>1111          196 CONFIGURATION FRAMES
0<DATA FRAME NUMBER 002>1111          EACH FIELD CONSISTS OF A
0<DATA FRAME NUMBER 003>1111          START ZERO FOLLOWED BY A
    |      |      |      |            71 DATA FIELD ENDING IN TWO
    |      |      |      |            OR MORE DUMMY BITS.
    |      |      |      |
    |      |      |      |
    |      |      |      |
    |      |      |      |

0<DATA FRAME NUMBER 195>1111
0<DATA FRAME NUMBER 196>1111
1111                                  MIMIMUM OF FOUR POST-
                                      AMBLE CODE BITS.
```

Figure 1. Data Stream Format

Mode Pin			Mode Selected
M0	M1	M2	
0	0	0	Master serial
0	0	1	Master low
0	1	1	Master high
1	0	1	Peripheral
1	1	1	Slave

Table 1. Five Configuration Modes Truth Table

Slave Mode of Configuration

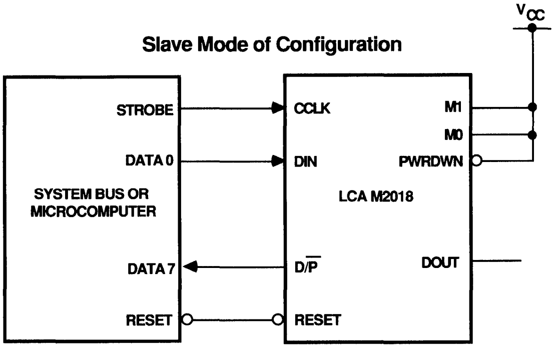


Figure 2. Configuration Diagram of Slave Mode

In the slave mode configuration, the data transfer occurs over a single data line, data 0 in this case. When data is valid the clock rising edge synchronizes the loading of data into the LCA device. The done/program pin may be monitored for end of configuration. This circuit may be used for microcomputers, microprocessors and interface adapters, but in this case the PC system bus is used.

PC CONFIGURATION HARDWARE

The LCA Applications Handbook (10098A) describes a number of hardware arrangements whereby configuration data may be downloaded serially to the LCA device. It is also supported with a download cable which is included with the XACT development system. The cable permits communication from a PC to the LCA through a parallel port. However, this method of downloading configuration data is not practical for an LCA device mounted on a card that is plugged into the IBM PC bus. This system uses the bus as a medium for configuration data transmission. Although the option for using the download cable has not been precluded from the interface described here. In the early stages of development it might be easier to use the download cable, until bus configuration is established. Figure 2 shows a block diagram of serial configuration mode while Figures 3 and 10 show the waveforms associated with this mode. In slave mode configuration the time to configure the LCA is determined by the loading source, in this case the controlling/loading program running in the PC.

Slave Mode Configuration Timing Considerations

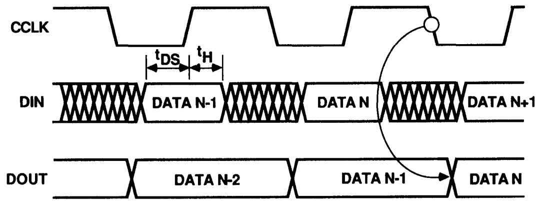


Figure 3. Slave Mode Timing

In the slave mode the CCLK pin is enabled as an input. Data set up at the DIN line is clocked into the LCA on the rising edge of the clock pulse. For the purposes of chaining LCA devices, data can be passed to the DOUT line but delayed by one clock cycle. Note that this data is synchronized to the falling edge of the clock. The configuration process is not complete until three additional clocks are appended to the pulse train. That is three more than the total number of configuration bits.

For all speed grades of the device the data setup and hold times are identical. A minimum figure of zero nanoseconds for setup and 40 nanoseconds for hold. The worst case figure for valid data at the DOUT pin is 65 nanoseconds after the falling edge of the clock.

Figure 7 is a schematic of the actual hardware used to drive the LCA device during configuration. The schematic shows PC bus signals involved in the process and the hardware used to decode them. A 74ALS520 decodes the PC address (set by SW1-SW7) for an I/O read/write cycle and a PAL16L8 decodes commands sent by download software (described later). Alternatively, a 74LS688 or equivalent could be used in place of the 74ALS520 but external pull-up resistors would be required on the Q inputs. Wired as shown, eight consecutive addresses will match as far as the 74ALS520 is concerned (e.g., 320H to 327H). The first of these eight addresses is termed the Base Port. Figure 8 shows the port addressing assignment. The port interface was required to work in a specific application that required a large decode range so the interface had a dual function. The port interface configuration was a biproduct of a real design.

Although the PAL16L8 (for which complete PAL design specification equations may be found in Appendix 7) is not a registered device, its outputs may be fed back such that latching can be achieved. This technique is used to store control information sent by configuration software. The C_0 and C_1 outputs from the PAL16L8 are latched condition codes where C_1 and C_0 store the data on D_7 and D_6, respectively, during an I/O write cycle to Base Port + 2. DPEN enables the D/~P PAL output and is latched from D5 on the same I/O write cycle.

A summary of the four condition codes is given in Table 2. For condition code 0, <C_1, C_0> = <0, 0>, the PAL device is in an idle mode. This is the mode preceding and following configuration. For condition code 1 the PAL asserts the the D/~P output as the trigger placing the LCA device in configuration mode. Condition code 2 directs the PAL to toggle -RESET for each subsequent write to Base Port + 3. This permits clearing of the LCA register contents. Finally, condition code 3 directs the PAL to toggle CCLK LOW, then HIGH, for each subsequent write to Base Port + 3. It is in this mode that configuration data bits are clocked into the LCA device, one bit at a time.

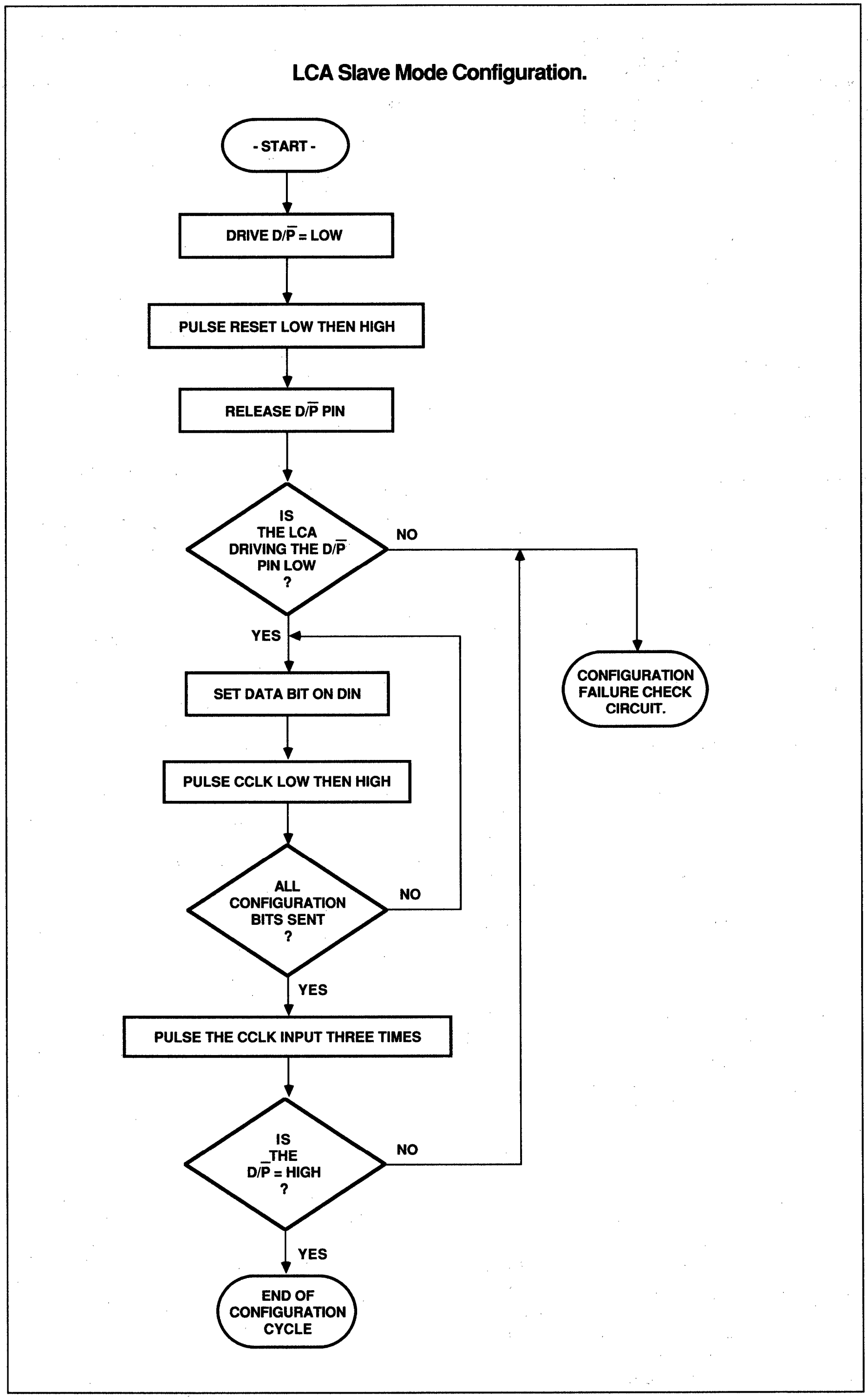

LCA Slave Mode Configuration.
- START -
DRIVE D/$\overline{P}$ = LOW
PULSE RESET LOW THEN HIGH
RELEASE D/$\overline{P}$ PIN
IS THE LCA DRIVING THE D/$\overline{P}$ PIN LOW ?
NO
YES
CONFIGURATION FAILURE CHECK CIRCUIT.
SET DATA BIT ON DIN
PULSE CCLK LOW THEN HIGH
ALL CONFIGURATION BITS SENT ?
NO
YES
PULSE THE CCLK INPUT THREE TIMES
IS _THE D/$\overline{P}$ = HIGH ?
NO
YES
END OF CONFIGURATION CYCLE

Figure 4. Flow Diagram of Configuration

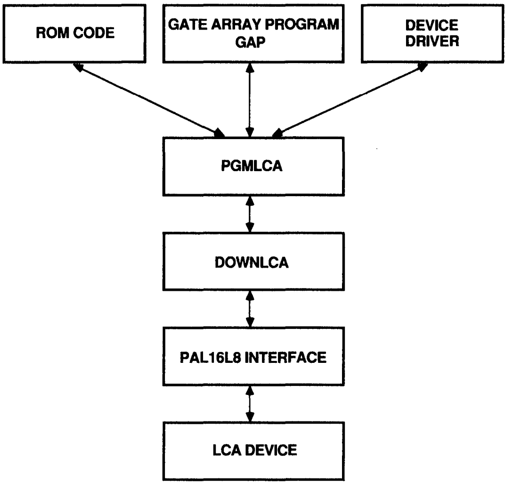


Figure 5. Flow Diagram of Software Packages, GAP

Condition Codes

C_0	C_1	Code	Function
0	0	0	Releases $\overline{D}$/P for for normal operation
1	0	1	Assert $\overline{D}$/P Low
0	1	2	Toggle RESET mode
1	1	3	Toggle CCLK mode

Table 2. PAL Condition Codes

After asserting D/~P (LOW) and toggling -RESET (condition codes 1 and 2) the software should read from Base Port + 2. During this read the PAL will drive the state of the D/~P onto bit 0 of the PC data bus. This is done through the DPOUT pin which is enabled only during reads from the I/O address Base Port + 3. An additional output, active LOW-LEN is also asserted at this time, allowing another on-board latch to drive data, such as switch settings, onto the bus. If the state of the D/~P pin is not 0 (LOW) then the LCA has failed to enter the configuration mode. Likewise, after sending the entire configuration bit stream and an additional three dummy bits (as required by the LCA) the software reads from Base Port + 2 to check that D/~P has been allowed to return HIGH, this indicates that the LCA has terminated the configuration process.

Timing considerations for this procedure are shown in Figures 9 and 10. Figure 9 shows a single I/O write cycle. The valid I/O address is put onto the bus, and decoded by the address decoder circuitry shown in Figure 7. The trailing (positive) edge of -IOW is used to latch condition bits. Also, in condition 2, CCLK follows -IOW during writes to Base Port + 3. Thus, the rising edge of CCLK corresponds to the rising edge of -IOW. This ensures that data will be latched by the LCA only when it is valid (write data is not guaranteed to be valid at the leading edge of -IOW).

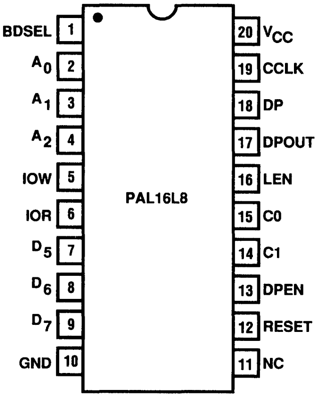


Figure 6. PAL16L8 Pinout Diagram

The C_0 and C_1 outputs from the PAL16L8 are latching outputs. Both C_0 and C_1 are addressed into the address baseport +2. The data lines $D_7$and D_6 are routed to C_1 and C_0 respectively during an I/O write operation. The one of four condition codes shown in Table 2 reflects the current status of the PAL during its interface activity with the LCA device.

As shown, SD0 on the PC bus is connected directly to DIN on the LCA through an in-line SPST switch. This switch should normally be closed, and need not be present at all in a production layout. It was added to allow the download cable, included with XACT, to function properly if required. If the switch is opened and the PAL removed, the download cable may be connected directly to the D/~P, -RESET, CCLK, and DIN pins and will interface directly through the cable.

BIT STREAM FORMAT AND CONVERSION

The format of the configuration bit stream used to program the LCA M2018 is shown in Figure 1. The bit stream starts with four dummy bits (1111), followed by a four-bit preamble code (0010), a 24-bit length count and another four dummy bits. The length count indicates the total number of bits in the bit stream, including the header just described. This number will be loaded into an internal counter in the LCA device allowing it to synchronize the loading of data during the configuration process. The actual configuration process is always serial, even during parallel configuration modes, internal serialization takes place in the LCA device.

The usual design entry technique uses the XACT design editor to create the logic design. The XACT conversion software then generates a bit pattern of the design that can be loaded directly into the LCA device. XACT can produce the bit stream in several different binary and ASCII formats. In this application the need to produce the bit stream in a format amenable to inclusion in an assembly language program dictated that the RAWBITS format be used. The RAWBITS output, placed in a file with the extension ".RBT," is ASCII. (Appendix 1 shows a portion of an ".RBT" file.) In general, one ASCII '0' or '1' is used to represent each bit of the configuration data. This data is not packed so it is not an efficient way of generating data for storage but it is suitable for loading into the LCA in real time.

Assembly language modules tend to define data in terms of bytes or words. For this reason, a Pascal program, BITCON.PAS, was written (see Appendix 2 for a complete listing). BITCON filters the ".RBT" file and produces an assembly language include file (given the extension ".BLB") which defines the configuration bit stream in terms of define byte directives acceptable to the Microsoft MASM™ assembler. This has the effect of packing the bit stream. For the sake of discussion the packed bit stream will be referred to as a byte stream. Additionally, BITCON counts the bits in the stream and the bytes used to hold the stream and defines two words at the beginning of the ".BLB" file for these quantities. A portion of a typical ".BLB" file may be found in Appendix 3. The net result is a structure which, once assembled into a main program, may be manipulated easily by machine language modules. Further, the assembled byte stream requires only 1.5K to 2K depending on the device being configured. Thus, the byte stream and associated software may easily be placed in a 4Kx8 or 8Kx8 ROM.

Hardware Configuration

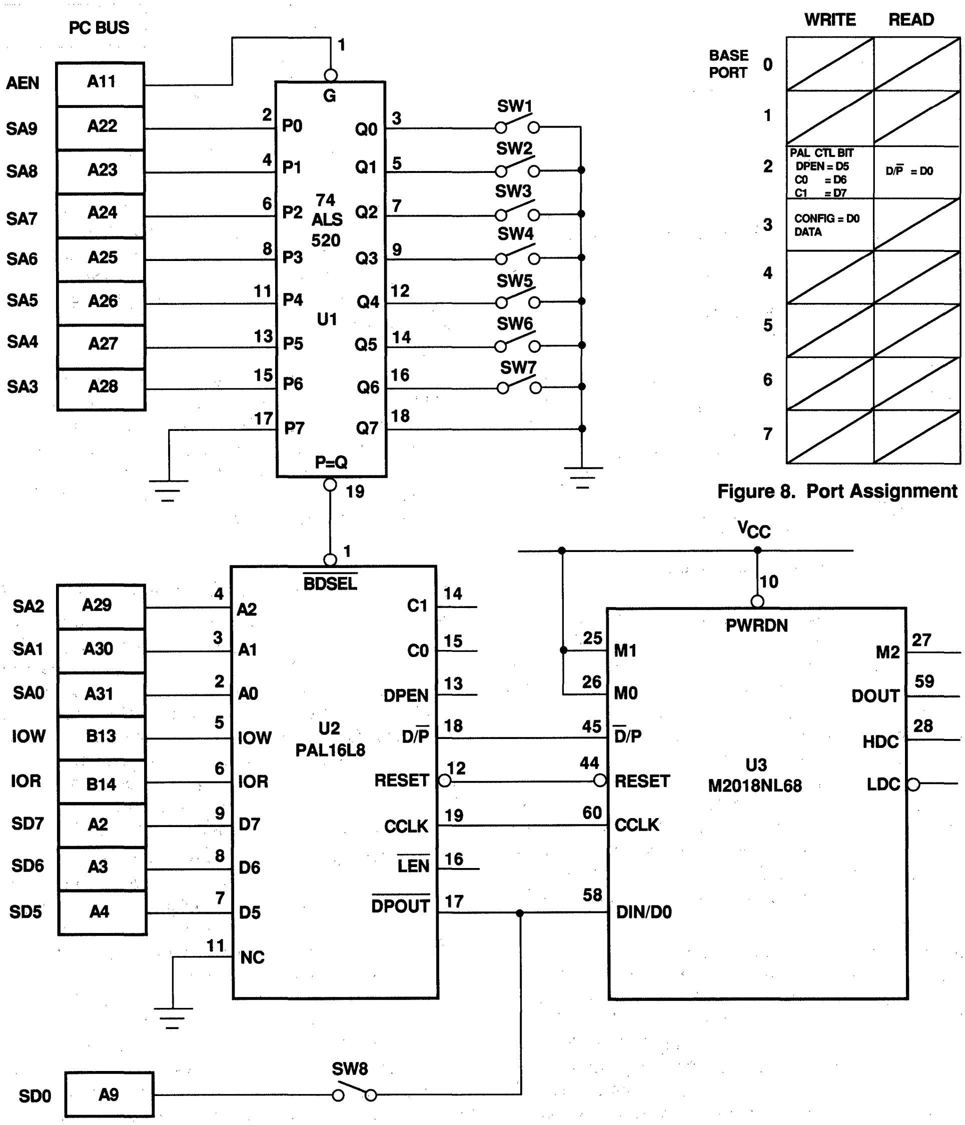


Figure 7. Circuit Implementation of PC BUS to LCA

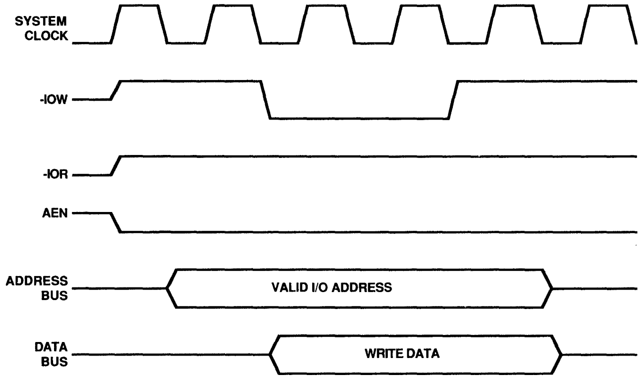
LCA Programming

BASE PORT	WRITE	READ
0		
1		

2	PAL CTL BIT
	DPEN = D5
	C0 = D6
	C1 = D7
		D/P̄ = D0

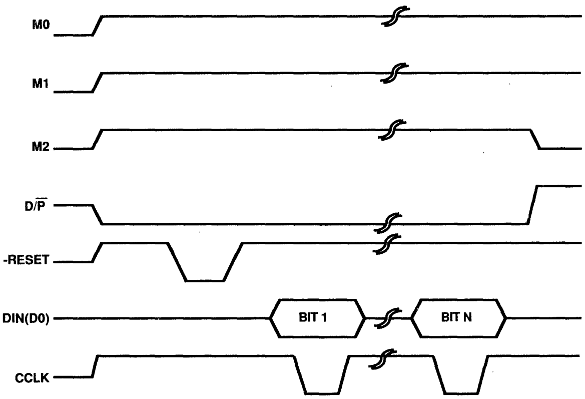
3	CONFIG = D0 DATA	
4		
5		
6		
7		

Figure 8. Port Assignment



Note: During bus write cycles, write data becomes valid after -IOW has been asserted. Data can be registered on the rising edge of -IOW.

Figure 9. Single WRITE Cycle. Waveform



Notes:
1. The CCLK must not remain low longer than 5 microseconds or LCA timing will be violated.
2. The D/P̄ pin is sampled after all the configuration bits + three additional bits have been sent. If the D/P pin is high then configuration was successful.

Figure 10. Configuration Cycle. Waveform

CONFIGURATION SOFTWARE

To perform the writing of configuration data to the LCA, the driver software was written in assembly language. High level languages such as 'C' or Pascal could be used, but assembly language was chosen because it is more efficient when considering the space constraints of a ROM. Three programs were written to interface to the LCA device;

GAP.ASM	Gate Array Program. See Appendix 4.
PGMLCA.ASM	Program LCA. See Appendix 5.
DOWNLCA.ASM	Download LCA. See Appendix 6.

Figure 4 shows a flow chart of a standard configuration cycle and Figure 5 shows the route taken from GAP to the programmed LCA device. The configuration driver may be partitioned into two halves: one generic and one hardware specific. The first half is implemented in a module called PGMLCA.ASM, which sequences through the configuration steps and decomposes the byte stream into the original configuration bit stream, while the second module DOWNLCA.ASM controls the bit manipulation required by the configuration hardware. Neither module carries the byte stream, it is assembled as part of a parent module whose job it is to invoke the actual configuration process. This is the function of a demonstration program GAP.ASM which displays some messages and calls the PGMLCA module, passing a pointer to the byte stream.

The byte stream is concatenated to the end of the GAP.ASM file as an include file. Any file that has been converted to byte stream ".BLB" can be downloaded to the LCA device by reassembling and linking GAP. The programs GAP.ASM, PGMLCA.ASM and DOWNLCA.ASM are written for Microsoft MASM version 5.0. A few utility routines in the library GAPLIB.LIB must also be linked to GAP but are not required by PGMLCA and DOWNLCA.

PGMLCA and DOWNLCA may just as easily be linked with a different parent and placed in ROM. This would be the production method for configuration used in the PC adapter card. The program in the ROM might perform a functional test at boot time to determine if the LCA is already programmed. If it is not, the PGMLCA module may be invoked by passing it a pointer to the byte stream. The entire process need only occur during the initial power on or cold boot, and takes less than a second to execute.

Looking at the modules in more detail, DOWNLCA interfaces to the PAL and provides three functions to PGMLCA. These include placing the LCA in the program mode, sending the LCA configuration bits and placing the LCA in the normal operation mode. As shown in the module listing, only a few port I/O instructions are used. The DOWNLCA program also checks to make sure that the LCA device enters and leaves the program mode at the correct times and reports this status to PGMLCA.

PGMLCA, sequences configuration by requesting program mode, sending configuration bits, and leaving the program mode. As discussed earlier, PGMLCA has the additional task of unpacking the byte stream that was originally packed by BITCON.

This partitioning allows the replacement of the DOWNLCA module with a different module suitable for supporting a different download hardware interface, without changing PGMLCA or BITCON. For example, the software for a download module to support the download cable of the development system could be generated. If the download cable is used, SW8 in Figure 7 must be switched to open.

CONCLUSION

Programming the LCA directly from the PC bus is a straightforward matter requiring a minimum of extra hardware. Once the hardware is breadboarded, the designer uses XACT to produce a RAWBITS file, which is converted by BITCON to an assembly language include file. This include file is then assembled as part of a parent program which is linked to the support modules PGMLCA and DOWNLCA. PGMLCA directs the configuration process through DOWNLCA which manages the specifics of the PAL-based interface.

The hardware interface may be easily adjusted to suit multiple LCA designs and the specific needs of other PC adapter cards or may be modified with a minimum of effort to support other microcomputer architectures. Likewise, BITCON is easily modified to produce include files for different assemblers. PGMLCA and DOWNLCA are also small modules which may be modified or ported as needed.

There are other advantages to this hardware and software scheme. Since the LCA configuration data is carried in software, which might be placed in ROM, an MS-DOS device driver, or even an application program, the realm of software updates is extended into the realm of hardware. At one end of the spectrum a vendor might supply hardware timing fixes or optimizations on diskette. At the other, the LCA could be designed as a generic logic block, whose function is dictated by the application software. In fact, at Trantor Systems a number of these capabilities are being exploited now in products incorporating such elements as RAM controllers and SCSI interfaces. Future LCA devices available from AMD, with even higher densities and faster speed grades will increase the scope of designs for which the part is appropriate.

SOFTWARE AVAILABILITY

The programs and PAL equations described in this application note, along with the binary library required for relinking GAP, are available on an MS-DOS 360K 5.25-inch diskette from:

Trantor Systems, Ltd.
33447 Western Avenue
Union City, CA 94587
(415) 489-3731

The charge per diskette is $25. Additional shipping charges may be necessary.

- APPENDIX 1 -

RAW BITS CONFIGURATION FILE.

```
XACT  LCA DESIGN.LCA 2018NL68
File design.rbt
22:15:17 DEC 10, 1987
22:15:17 DEC 10, 1987
Source
Version
Produced by XACT version 1.30
111100100000000001000101110101101111
0111111111110100111111011111111011111111111111110111101100111111100111111001111111111111111111
0110110111111111011111110111111111001111111111111101110011101110111011111111011111111111111111
                                        •
                                        •
                                        •
                      192 LINES OF RAW BIT CONFIGURATION DATA
                                        •
                                        •
                                        •

0001111101111011011110110111110010111110101011111110011111110011110100111101001111010111111
0111011111110111011101111111111111111111111111111111111111111111111111111111111011101111111
1111
```

Appendix 1. Raw Bit Listing

```
{
                          APPENDIX 2

  BitCon.pas

  BitCon is a bit conversion program which takes as its input the
  "*.rbt" file produced by the makebits command in the gate array
  development system and produces an assembly language file defining
  the bit stream.

  The "*.rbt" file contains a text prolog indicating the design name
  and the time/date of creation. The file then contains a number
  of lines containing ascii '1's and '0's which represent the bit
  stream. The first line of this stream contains the bit stream
  prolog, '1111', which BitCon uses to identify the bit stream.

  BitCon packs the bit stream into bytes where the highest order
  bit in a byte corresponds to the first encountered bit in the
  bit stream and the first byte corresponds to the first eight
  bits of the bit stream, and so forth.

  The output file has the form:

    ; text prolog created by makebits

      dw  length_of_bit_stream_in_bits
      dw  length_of_bit_stream_in_bytes

      db  byte_0,byte_1,...,byte_7
      db  byte_8,byte_9,...,byte_15
      ...
      ...
      db  ...,byte_n

  If the number of bits is not an even multiple of eight then then
  only the highest order bits of the last byte will be valid.

  The output file has the filetype ".blb" (for Bit LiBrary) and may
  be included in an assembly language program for assembly.

  history:

    12-09-87  RCB  first cut
    12-11-87  RCB  emit ';' instead of ':' in third line of
                   our portion of the prolog (1.0b)
    12-12-87  RCB  place the number of bits we counted into the
                   24-bit length field of the stream to ensure that
                   the two are the same (1.0c)
    01-04-88  RCB  make release version for ap. note (1.1a)
    01-26-88  RCB  more of mod. on 01-04-88 (1.1b)
}


const
  title = 'BitCon: .rbt -> .blb Filter, Version 1.1b';
  copyright = 'Copyright (c) 1987, Trantor Systems, Ltd.';

  max_byte = 8191;            { max bytes in byte stream - 1 }
  stream_prolog = '1111';     { ascii form of bit stream prolog }
  max_bytes_per_line = 8;     { max bytes defined per line in output file }

  intype = '.rbt';            { input filetype }
  outtype = '.blb';           { output filetype }


type
  st = string [132];


var
  byte_stream: array [0..max_byte] of byte;
  bit_length: integer;
  byte_length: integer;

  rbt_file: text;
  blb_file: text;
  current_line: st;


{ convert the byte argument to a three-digit hex number (includes leadin
  zero). also add a trailing 'h'. }

function hex (value: byte): st;

function nibble (value: byte): st;
begin
  nibble := copy ('0123456789abcdef', value + 1, 1);
end;

begin
  hex := '0' + nibble (value div 16) + nibble (value mod 16) + 'h';
end;


{ if the filename doesn't have a filetype, append the specified one. if
  there is a filetype, replace it with the specified one. }

function fnm (filename: st; filetype: st): st;
begin
  if pos ('.', filename) = 0 then
    fnm := concat (filename, filetype)
  else
    fnm := concat (copy (filename, 1, pos ('.', filename) - 1), filetype
end;


{ copy lines from the prolog of the .rbt file to the prolog of the .blb
  file until the bit stream prolog is found. leave the first line of the
  bit stream in current_line. }
```

```
procedure read_prolog;
begin
  readln (rbt_file, current_line);
  while (not eof (rbt_file)) and
        (copy (current_line, 1, length (stream_prolog)) <> stream_prolog
    writeln (blb_file, ';', ^I, current_line);
    readln (rbt_file, current_line);
  end;
  writeln (blb_file);
  writeln (blb_file, ';', ^I, 'Translated by:');
  writeln (blb_file, ';', ^I, title);
  writeln (blb_file, ';', ^I, copyright);
  writeln (blb_file);
end;


{ read the bit stream and construct a byte stream in memory. keep a count
  of the number of bits and the number of bytes. we assume that the first
  line of the bit stream is already in the variable current_line.}

procedure read_bit_stream;
var
  cur_bit: byte;
  their_length: integer;

procedure pack_bits;
var
  char_index: integer;

begin
  for char_index := 1 to length (current_line) do begin
    bit_length := bit_length + 1;
    if cur_bit = 7 then
      byte_stream [byte_length] := 0;
    byte_stream [byte_length] := byte_stream [byte_length] or
      ((ord (current_line [char_index]) - ord ('0')) shl cur_bit);
    if cur_bit > 0 then
      cur_bit := cur_bit - 1
    else begin
      cur_bit := 7;
      byte_length := byte_length + 1;
    end;
  end;
end;

begin
  bit_length := 0;
  byte_length := 0;
  cur_bit := 7;
  pack_bits;
  while not eof (rbt_file) do begin
    readln (rbt_file, current_line);
    pack_bits;
  end;
  if cur_bit <> 7 then
```

```
    byte_length := byte_length + 1;
  their_length := byte_stream [3] + (byte_stream [2] shl 8);
  if their_length <> bit_length then begin
    writeln ('Fixing up bit stream length...');
    writeln ('Old length = ',their_length,' bits');
    byte_stream [3] := bit_length and $ff;
    byte_stream [2] := (bit_length shr 8) and $ff;
  end;
end;


{ write_byte_stream formats the byte_stream just created by read_bit_str
  a bit length count and byte length count preceed the byte stream. }

procedure write_byte_stream;
var
  count: integer;

begin
  writeln (blb_file, ^I, 'dw', ^I, bit_length, ^I, ';# of bits in stream
  writeln (blb_file, ^I, 'dw', ^I, byte_length, ^I, ';# of bytes to hold
  for count := 0 to byte_length - 1 do begin
    if count mod max_bytes_per_line = 0 then begin
      writeln (blb_file);
      write (blb_file, ^I, 'db', ^I);
    end;
    if count mod max_bytes_per_line <> 0 then
      write (blb_file, ', ');
    write (blb_file, hex (byte_stream [count]));
  end;
  writeln (blb_file);
  writeln (blb_file);
end;


begin { main }
  writeln (title);
  writeln (copyright);
  if paramstr (1) = '' then begin
    writeln;
    writeln ('Usage: bitcon bitfile');
    writeln;
    writeln ('Where "bitfile" is a ".rbt" file produced by');
    writeln ('makebits. BitCon will produce an output file,');
    writeln ('"bitfile.blb," which contains an assembly');
    writeln ('language representation of the bit stream.');
    writeln;
  end
  else begin
    writeln;
    assign (rbt_file, fnm (paramstr (1), intype));
    {$i-}
    reset (rbt_file);
    if ioresult <> 0 then begin
      writeln ('Unable to open file "',fnm (paramstr (1), intype), '".')
      halt;
    end;
    writeln ('Reading "', fnm (paramstr (1), intype), '".');
    assign (blb_file, fnm (paramstr (1), outtype));
    rewrite (blb_file);
    if ioresult <> 0 then begin
      writeln ('Unable to create file "',fnm (paramstr (1), outtype), '"
      halt;
    end;
    {$i+}
    writeln ('Copying prolog...');
    read_prolog;
    if not eof (rbt_file) then begin
      writeln ('Reading bit stream...');
      read_bit_stream;
      writeln ('Bit stream = ',bit_length, ' bits');
      writeln ('           = ',byte_length, ' bytes');
      writeln ('Writing bit stream...');
      write_byte_stream;
    end
    else
      writeln ('Input file ended prematurely.');
    close (blb_file);
    close (rbt_file);
    writeln ('File "',fnm (paramstr (1), outtype),'" produced.');
  end;
end.
```

- APPENDIX 3 -

LISTING OF FILE AFTER CONVERSION BY BITCON

```
;     XACT   LCA  DESIGN.LCA 2018NL68
;     File design.rbt
;     22:15:17 DEC 10,  1987
;     22:15:17 DEC 10,  1987
;     Source
;     Version
;     Produced by XACT version 1.30

;     Translated by:
;     bitcon: .rbt -> .blb filter, version 1.0b
;     Copyright (c) 1987, Trantor Systems, Ltd.

      dw     17876   ;# of bits in stream
      dw     2235    ;# of bytes to hold stream

      db     0f2h, 000h, 045h, 0d4h, 0f7h, 0ffh, 04fh 0dfh
      db     0efh, 0ffh,  0efh,  067h, 0f3h, 0f3h,0ffh, 0feh

                     ●                    ●
                     ●                    ●
                     ●                    ●

               193 LINES OF COMPRESSED CODE

                     ●                    ●
                     ●                    ●
                     ●                    ●

       db        0bbh,  0bfh,  0ffh,   0ffh,   0ffh,   0ffh,   0ffh,   0ffh
       db        0eeh,  0ffh,  0f0h
```

Appendix 3. Listing of <file>.blb

```
        page    62,132
        title   GAP, Gate Array Programmer
;----------------------------------------------------------------------
;
;       APPENDIX 4
;
;       GAP.asm
;
;       GAP takes the byte stream generated by BitCon (which, in turn,
;       took its input from MAKEBITS in the LCA development system) and
;       downloads it to the gate array. GAP itself only displays signon
;       messages and statistics. It then passes a pointer to a "byte
;       stream" (a packed configuration bit stream) which the PGMLCA
;       module decodes.
;
;       GAP is only intended for debugging and demonstration. The gate
;       array programming modules, on the other hand, are intended for
;       production use. Invoke GAP with:
;
;               A>gap
;
;       Invoking GAP without the hardware described in the ap. note
;       shouldn't have any ill effect. However, be careful that no
;       other card occupies port address 328h-32fh for which this demo.
;       has been "hard-wired."
;
;       MASM 5.0 may be used to assemble this module. It must then be
;       linked to the other modules.
;
;       History:
;
;               12-11-87  RCB   first (1.0a)
;               01-26-88  RCB   make mods. for ap. note release (1.1a)
;
;       Copyright (C) 1988, Trantor Systems, Ltd.
;       All rights reserved.
;
;----------------------------------------------------------------------

        page
;----------------------------------------------------------------------
;       Externals and Defs.
;----------------------------------------------------------------------

;       pgm_gate_array is an entry in the pgmlca module. this routine
;       sequences the programming phases.

        extrn   pgm_gate_array:near

;       dsp_str and dsp_num are utility routines to display an ASCIIZ
;       string and a decimal number, respectively.

        extrn   dsp_str:near
        extrn   dsp_num:near
```

```
prog_title               equ     'GAP: Gate Array Programmer, version 1.1
prog_copyright           equ     'Copyright (c) 1987, Trantor Systems, Lt

false                    equ     0
true                     equ     not false
cr                       equ     0dh
lf                       equ     0ah
eof                      equ     1ah
null                     equ     0
dos                      equ     21h
dos_program_terminate    equ     4ch
combase                  equ     100h

ok_ret_code              equ     0               ;program return codes
bad_ret_code             equ     1

        page
;---------------------------------------------------------------------
;       COM file header and program entry
;---------------------------------------------------------------------

code    segment para public
        assume  cs:code,ds:code,es:code,ss:code

                org     combase
start:          jmp     main
signon:         db      cr,prog_title,cr,lf
                db      prog_copyright,cr,lf,lf,null,eof

                dw      256 dup (?)
stack:


main    proc

        mov     ax,cs
        mov     ds,ax
        mov     es,ax
        cli
        mov     ss,ax
        lea     sp,stack
        sti

        lea     dx,signon               ;signon
        call    dsp_str
        lea     dx,pgm_msg              ;now programming...
        call    dsp_str

        lea     di,gate_array_code      ;di -> bit stream structure

;       the bit stream structure consists of 3 fields: a word count
;       of the bits in the bit stream, a word count of the bytes needed
;       to store the bit stream, and an array of bytes of the length
;       indicated by the previous (second) field.
```

```
        mov     ax,[di]                 ;show bit length
        call    dsp_num
        lea     dx,bit_msg
        call    dsp_str
        mov     ax,[di + 2]             ;show byte length
        call    dsp_num
        lea     dx,byte_msg
        call    dsp_str

        call    pgm_gate_array          ;es:di -> bit stream structure
                                        ;cy indicates error upon return
        lea     dx,ok_pgm_msg
        mov     al,ok_ret_code
        jnc     done
        lea     dx,bad_pgm_msg          ;show error
        mov     al,bad_ret_code
done:
        call    dsp_str

        mov     ah,dos_program_terminate        ;al is still the return
        int     dos

main    endp

pgm_msg:        db      'Programming gate array...',cr,lf,null
bit_msg:        db      ' bits in stream.',cr,lf,null
byte_msg:       db      ' bytes in stream.',cr,lf,null
ok_pgm_msg:     db      'Gate array programmed.',cr,lf,null
bad_pgm_msg:    db      'Programming failure.',cr,lf,null

        page
;-----------------------------------------------------------------------
;       "Byte stream"
;
;       The "*.BLB" file containing the "byte stream" (packed
;       configuration bit stream) which was produced from the "*.RBT"
;       file bit BitCon is included here.
;-----------------------------------------------------------------------

gate_array_code:        include dummy.blb

code    ends
        end     start

;       end of file
```

```
        page    62,132
        title   DOWNLCA, LCA download routines (using PAL)
;-----------------------------------------------------------------------
;
;       APPENDIX 5
;
;       DOWNLCA.asm
;
;       DOWNLCA contains the hardware specific code for downloading
;       a configuration bit stream to the LCA using the Trantor
;       configuration PAL interface.
;
;       The PAL interface gives us direct control over the D/-P,
;       -RESET, and CCLK pins on the LCA.
;
;       To place the LCA in program mode we set the D/-P low and
;       toggle reset. We then release the D/-P pin and check to see
;       that the LCA is still driving it low, signifying that it
;       entered program mode.
;
;       The CCLK pin is normally held high and is pulsed low as
;       each configuration bit is sent. Each configuration bit is
;       sent on SD0 (bit 0 of the PC's data bus). Thus, the rising
;       edge of CCLK occurs at the end of the write cycle. This is
;       necessary as the PC doesn't drive write data until after the
;       leading edge of the write signal.
;
;       After the configuration bit stream has been sent we check
;       to see if the LCA allowed D/-P to go high, signifying the
;       end of configuration.
;
;       This module may be placed in rom if so desired.
;
;       History:
;
;               12-11-87  RCB   first
;               01-26-88  RCB   make mods. for ap. note release
;               04-11-88  RCB   release dp pin after toggling
;                               reset
;
;       Copyright (C) 1988, Trantor Systems, Ltd.
;       All rights reserved.
;
;-----------------------------------------------------------------------

        page
;-----------------------------------------------------------------------
;       Publics and Defs.
;-----------------------------------------------------------------------

        public  program_lca             ;program/normal entry
        public  send_lca_bit            ;send bit entry

;       LCA programming codes passed between caller and program_lca
;       entry point.
```

```
lca_func_normal         equ     0
lca_func_program        equ     1

srl_rsc_dta             equ     3       ;rsc data port (bi-directional)
srl_dta_port            equ     3       ;board data port (out)

;       This PAL configuration interface was designed into an existing
;       port mapping scheme. For demo purposes, the base port to which
;       the card containing the LCA responds is defined as an equate.
;       The ports which control configuration are defined as offsets
;       from this base port.

base_port               equ     328h
config_ctl_port         equ     base_port + 2
config_dta_port         equ     base_port + 3

;       Control port bit assignments. The configuration PAL latches
;       c1, c0, and dpen during writes to the ctl port. The config. PAL
;       drives data onto SDO during reads from this port.

ctlbit  record  c1:1,c0:1,dpen:1,clt_unused:5
statbit record  stat_unused:7,dp:1

;       Control bits c1 and c0 set PAL modes. The function implied
;       by a given mode doesn't take effect until a subsequent write
;       to the config_dta_port (this scheme eliminates spiking on
;       the control lines).

mode_idle       equ     0                       ;configuration PAL idle
mode_program    equ     mask c0 + mask dpen     ;set pgm mode
mode_reset      equ     mask c1 + mask dpen     ;toggle reset
mode_reset_only equ     mask c1                 ;   "     "   without dp
mode_cclk       equ     mask c1 + mask c0       ;send configuration bits

        page
;------------------------------------------------------------------------
;       module entry
;------------------------------------------------------------------------

code    segment byte public
        assume  cs:code

;------------------------------------------------------------------------
;       program_lca
;
;       input:  al = program/normal function code
;
;       output: nc = function ok
;               cy = function failed
;
;       we support two functions: set program mode, set normal mode.
;       when we go to program mode we also pulse the lca's reset.
;       when going to normal mode we check to make sure that the lca
;       has allowed d/-p to go high, indicating programming finished.
```

```
;
;       saves all registers
;-----------------------------------------------------------------------

program_lca     proc

        push    ax
        push    dx

        mov     dx,config_ctl_port          ;dx -> srl board control port

        cmp     al,lca_func_normal          ;go to normal operation?
        jz      go_normal

;       Program mode
;
;       1. set program mode
;       2. toggle reset
;       3. check status of d/-p
;       4. go to cclk mode

        mov     al,mode_program             ;set program mode and assert d/-
        out     dx,al
        inc     dx                          ;dx -> config data port
        out     dx,al                       ;any write latches program mode
        dec     dx
        mov     al,mode_reset               ;assert d/-p and enable -reset
        out     dx,al
        inc     dx
        out     dx,al                       ;any write here toggles -reset
        dec     dx

        mov     al,mode_reset_only          ;stay in reset mode but stop
                                            ;driving d/-p
        out     dx,al

        in      al,dx                       ;get status of d/-p pin
        test    al,mask dp                  ;bit should be low if in pgm mod
        stc
        jnz     program_done

        mov     al,mode_cclk                ;enable cclk for configuration b
        out     dx,al
        clc
        jmp     short program_done

;       Normal mode
;
;       1. set PAL to idle mode
;       2. check d/-p to make sure lca took it high
;       3. also output a zero to base port to disable all funcs.

go_normal:
        mov     al,mode_idle                ;shut-down programming PAL
        out     dx,al
        inc     dx                              ;dx -> board dta port
        out     dx,al                           ;any write shuts down PA

;       We write a 0 to the base_port to disable all board functions.
;       This is necessitated by the particular board and not by the
;       demands of the configuration architecture.

        mov     dx,base_port
        xor     al,al
        out     dx,al

;       check the d/-p pin

        mov     dx,config_ctl_port
        in      al,dx                   ;get board status
        test    al,mask dp              ;check d/-p, should be high
        jnz     program_done
        stc

program_done:
        pop     dx
        pop     ax
        ret

program_lca     endp

;----------------------------------------------------------------------
;       send_lca_bit
;
;       input:  al, bit 0 = bit of configuration data
;
;       output: nc = bit sent ok
;               cy = error
;
;       send a bit of configuration data to the lca.
;       this routine must not be called unless the lca has been placed
;       in program mode by the program_lca routine.
;
;       saves all regs.
;----------------------------------------------------------------------

send_lca_bit    proc

        push    dx

        mov     dx,config_dta_port
        out     dx,al                   ;send the bit, the pal
                                        ;will toggle cclk

        clc                             ;for now we have no handshake
                                        ;on each bit so we can't determi
                                        ;whether or not an error occurre

        pop     dx
        ret

send_lca_bit    endp

code    ends
        end

;       end of file
```

```
        APPENDIX 6

        page    62,132
        title   PGMLCA: Gate array programming module
;-----------------------------------------------------------------------
;
;       PGMLCA.asm
;
;       PGMLCA sequences the phases of LCA programming. In general
;       these phases include:
;
;               o       place LCA in program mode, toggle reset
;               o       check that the LCA entered program mode
;               o       send the configuration bit stream, one
;                       bit at a time
;               o       place the LCA in normal mode
;               o       check that the LCA is in normal mode
;
;       PGMLCA may be placed in ROM, a DOS device driver, or other
;       program. The caller passes a far pointer to a "byte stream",
;       a packed configuration bit stream. The byte stream has the
;       structure:
;
;               word            # of bits in stream (N)
;               word            # of bytes used to store stream (M)
;               byte [0..M-1]   M bytes holding packed bit stream
;
;       PGMLCA uses another module, DOWNLCA, to perform hardware-
;       dependent functions such as sending bits and setting program/
;       normal mode.
;
;       History:
;
;               12-11-87  RCB   first
;               01-26-88  RCB   make mods. for ap. note release
;
;       Copyright (C) 1988, Trantor Systems, Ltd.
;       All rights reserved.
;
;-----------------------------------------------------------------------

        page
;-----------------------------------------------------------------------
;       Publics, Externals, and Defs.
;-----------------------------------------------------------------------

        public  pgm_gate_array          ;our entry

;       program_lca places the LCA in program/normal mode and
;       reports the success or failure of the function. send_lca_bit
;       sends individual bits to the LCA.

        extrn   program_lca:near
        extrn   send_lca_bit:near
```

```
;       lca programming codes passed between us and the program_lca
;       entry point.

lca_func_normal         equ     0
lca_func_program        equ     1

        page
;-----------------------------------------------------------------------
;       module entry (pgm_gate_array)
;
;       input:  es:di -> bit stream structure
;                        es:[di] = word, bit stream length in bits
;                        es:[di+2] = word, bytes used to hold stream
;                        es:[di+4] -> array of bytes holding stream
;
;       output: nc = programming ok
;               cy = unable to program lca
;
;       saves all regs
;-----------------------------------------------------------------------

code    segment byte public
        assume  cs:code

pgm_gate_array  proc

        push    ax
        push    bx
        push    cx
        push    di

        mov     al,lca_func_program     ;set lca to programming mode
        call    program_lca
        jc      pgm_error

        mov     cx,word ptr es:[di]     ;cx = number of bits to send
        add     di,4                    ;es:di -> byte array
pgm_byte:
        mov     bl,8                    ;8 bits/byte
        mov     al,byte ptr es:[di]     ;al = packed array of bits
                                        ;bit 7 is first to be sent
        inc     di
pgm_bit:
        jcxz    pgm_done
        dec     cx
        rol     al,1                    ;al, bit 0, is bit to send
        push    ax
        and     al,1                    ;mask bit
        call    send_lca_bit            ;and send
        pop     ax
        jc      pgm_error               ;couldn't send bit
        dec     bl
        jnz     pgm_bit
        jmp     pgm_byte
```

```
pgm_done:

;       the LCA requires that three additional bits be sent after
;       the configuration bit stream in order to complete configuration.

        call    send_lca_bit
        call    send_lca_bit
        call    send_lca_bit

        mov     al,lca_func_normal      ;set lca to normal operation
        call    program_lca
        jnc     pgm_finished

pgm_error:
        stc                             ;set error return code

pgm_finished:
        pop     di
        pop     cx
        pop     bx
        pop     ax
        ret

pgm_gate_array  endp

code    ends
        end

;       end of file
;               APPENDIX 7
;
title           LCA Programming PAL
pattern         config
revision        B
author          Robert Botchek
company         Trantor Systems, Ltd
date            January 26, 1988
chip            config pal1618

bdsel a0 a1 a2 iow ior d5 d6 d7 gnd
nc /reset /dpen /c1 /c0 /len /dpout /dp /cclk vcc


equations

; c1 and c0 are latching outputs which are used as function selects.
; they are latched during a write to baseport+2. the meaning of the
; function selects are defined:
;
;       c1 c0   code    function
;       -- --   ----    --------
;       0  0    0       releases d/-p for normal lca operation
;       0  1    1       assert lca program (d/-p low)
;       1  0    2       toggle -reset mode
;       1  1    3       toggle cclk mode
;
; c1 and c0 will glitch during latching and so should not be used
; as direct inputs to a combinatorial output.

c1              =       /bdsel * /a2 * a1 * /a0 * /iow * d7
                +       bdsel * c1
                +       a2 * c1
                +       /a1 * c1
                +       a0 * c1
                +       iow * c1
c1.trst         =       vcc

c0              =       /bdsel * /a2 * a1 * /a0 * /iow * d6
                +       bdsel * c0
                +       a2 * c0
                +       /a1 * c0
                +       a0 * c0
                +       iow * c0
c0.trst         =       vcc

; dpen latches similarly to c1 and c0. when asserted it enables the
; dp pin.

dpen            =       /bdsel * /a2 * a1 * /a0 * /iow * d5
                +       bdsel * dpen
                +       a2 * dpen
                +       /a1 * dpen
                +       a0 * dpen
                +       iow * dpen
```

```
dpen.trst       =       vcc

; dp drives the d/-p input on the lca. when at vcc, the lca operates
; normally (assuming it has been programmed). when at gnd, the lca
; is in a program mode.
;
; d/-p is latched during functions 1 and 2 of (<c1,c0> = <0,1> or
; <c1,c0> = <1,0>) and any write to baseport+3 is given. an assertion
; of d/-p also enables the d/-p output.

dp              =       /bdsel * /a2 * a1 * a0 * /iow * /c1 * c0
                +       /bdsel * /a2 * a1 * a0 * /iow * c1 * /c0
                +       bdsel * dp
                +       a2 * dp
                +       /a1 * dp
                +       /a0 * dp
                +       iow * dp
dp.trst         =       dpen

; reset drives the -reset line on the lca. it is asserted only during
; a write to baseport+3 when function code 2 is selected.

reset           =       /bdsel * /a2 * a1 * a0 * /iow * c1 * /c0
reset.trst      =       vcc

; cclk is the configuration clock used by the lca to latch incoming
; configuration data. the lca latches data on the rising edge of cclk.
; we keep cclk high all the time and only lower it during a write to
; baseport+3 when function code 3 is selected. thus, at the end of
; the write cycle the data will be latched.
;
; note: data is written to baseport+3, not baseport+2. this simplifies
; the timing of the cclk enable and the data output.

cclk            =       /bdsel * /a2 * a1 * a0 * /iow * c1 * c0
cclk.trst       =       vcc

; len enables the board status latch onto the pc's data bus, it also
; enables the dpout output (below).
;
; board status includes the condition of the lca's done/-program
; (d/-p) pin, etc.

len             =       /bdsel * /a2 * a1 * /a0 * /ior
len.trst        =       vcc

; dpout drives data onto the pc's data bus during a read from base
; port+2 as enabled by len. dpout indicates program mode when high.

dpout           =       /dp
dpout.trst      =       len
```

LCA™ Video Controller Design from OrCAD™ Schematic Design Tool

Chris Jay and Karen Spesard

The Logic Cell™ Array (LCA device) is a programmable logic circuit that can require a high level of design capability from digital systems designers. Fortunately, system support is available from AMD in the form of CAE software, much of it PC® based, which makes the task of logic design for the LCA device a relatively straight forward procedure. This provides a benefit to the designer, allowing the creation of complex VLSI functions in comparatively short design cycles, thus reducing the overall design cycle and improving time to market.

The OrCAD schematic capture program and interface conversion software allow the designer to use diagramatic entry on the PC/AT. This makes logic design within the LCA less product specific and more general because the designer does not need to know much detail about the LCA architecture. This is especially useful to those designers who do not have a background in designing with programmable logic devices, PLDs and are unfamiliar with software packages that support Boolean and state machine design entry. Also, the schematic capture package provides an output that can be used for documentation, giving a range of standard symbols for gates and register logic varying in degrees of complexity up to counter and shift register circuits. This application note highlights the design of a video controller circuit in the M2064 LCA device using the OrCAD SDT/III schematic capture package, support libraries and interface software.

10471A
JULY 1988

LCA™ Video Controller from OrCAD™ Schematic Design Tool

by Chris Jay and Karen Spesard

Introduction

The usual design entry for the LCA is through a dedicated software package called XACT™. The XACT software is product specific in the respect that its most effective use relies on the logic designer understanding the LCA device architecture. Once a design has been created in the XACT environment it may be converted to a bit pattern suitable for programming into a PROM. This is the configuration data for the LCA device and may be loaded into the LCA during its configuration sequence. Once the LCA has read all the configuration data bits, it enters a functional mode and performs the logic function intended by the designer. The route from design entry in XACT to the final design has been enhanced with the addition of the OrCAD schematic entry. Figure 1 shows a flow diagram highlighting the route taken from schematic entry into the XACT editor. The schematic capture package OrCAD SDT/III is supported with a compatable library of symbols and macro schematics. Any design created without this support library cannot be converted into an XACT compatable design, it must be redrawn using the support libraries.

The schematic file generated in OrCAD has the extension .SCH which is the file containing graphic information in a binary format. The OrCAD to LCA interface operates on the .SCH file generating an XNF file. This is the eXternal Netlist Format file containing all the component and net list information.

The XNF2LCA program converts the XNF file to an LCA file. Although the design is not routed into the LCA, logic reduction and partitioning has taken place and the CLBs have been located in the device. At this point the XACT conversion exists but no interconnection has been made. The designer may enter the XACT environment at this stage and route the design using the ROUTE command in XACT. The auto route program will connect the CLBs according to the NETLIST information but CLB placement will not be rearranged.

The Automatic Design Implementation software contains an Auto Place and Route program, APR which will operate on the .LCA file, placing CLBs and interconnecting them. It is possible that a route for some of the nets will not be found. In this case the software will indicate how many nets it has failed to route. The design may be transferred to the XACT environment and manual routing may be used or the APR software can be rerun. Once the design is in XACT the MAKEBITS and MAKEPROM option can be invoked to create the configuration file for the LCA device.

AMD supplies the complete engineering support software as a bundle, LCA-MDS151. Rather than buy individual software packages the user can purchase the cost-effective system bundle which consists of the OrCAD/SDT III schematic editor, OrCAD LCA Design Libraries and Interface, XACT Design Editor and Automatic Design Implementation.

System Configuration

The LCA device is programmed to perform the necessary control functions for a small system video controller. The device was designed to provide video outputs for frame synchronization, line synchronization and three serial data outputs for red, green and blue video data streams. The internal address counters for video RAM and character memory are MA0–MA11 and R0–R3, respectively. Figure 2 shows the system configuration of the LCA device, support memory and interface circuits. The video RAM address outputs can be configured to the memory through bus driver devices and put into a 3-State condition so the host system can update the data to be displayed through a bus driver device 74LS244. In the final design the host system could interrogate the current status of the video controller for active line and frame synchronization. If active the video screen is blanked and the host system can update video data in RAM.

Parallel inputs for DR0–DR7, DG0–DG7 and DB0–DB7 enable data to be loaded from the character memory into the three internal shift registers. A high speed counter driven from an internal crystal controlled oscillator (circuit details are shown in Appendix 2) supervises one load and seven shift states for all three video shift registers and, as a consequence, determines the 'dot rate' of the video data output. Figure 3 shows a block diagram of the design as it was configured in the LCA device. The crystal oscillator drives a high speed eight state counter. Registers QA, QB and QC form a modified linear feedback shift register designed for high speed clocking. The register outputs are decoded to control the load and shift operations in the video shift registers. After one load and seven shifts, the line counter and synchronizing circuitry are incremented to access the next character in video RAM which is waiting to be displayed. Figure 4a shows the load control waveform in relation to the system clock. The rising edge of the parallel load output is used to increment the memory address counters MA0–MA6 and, 400 nano seconds later, to load the internal video shift registers with dot data accessed during that current cycle. An additional internal shift register is used to delay the parallel load signal by three dot clock cycles to allow the current video data to be accessed and set up at the input to the 74LS374 octal register. There is a delay of 150 nano seconds before the rising edge of the delayed load output, giving time to access data in the video RAM and to satisfy the set up requirements of the 74LS374. To allow data in the character memory to be adequately set up at the input to the video shift registers, a further 250 nano seconds will elapse before the rising edge of the next parallel load output. A detailed description of how video data is addressed, decoded and displayed is given in Appendix 1. The MA0–MA6 outputs are clocked by an internal line counter to access the current column of video data from the video RAM. This address is incremented after each video shift and load operation. After one complete line scan, the counter is reset to commence

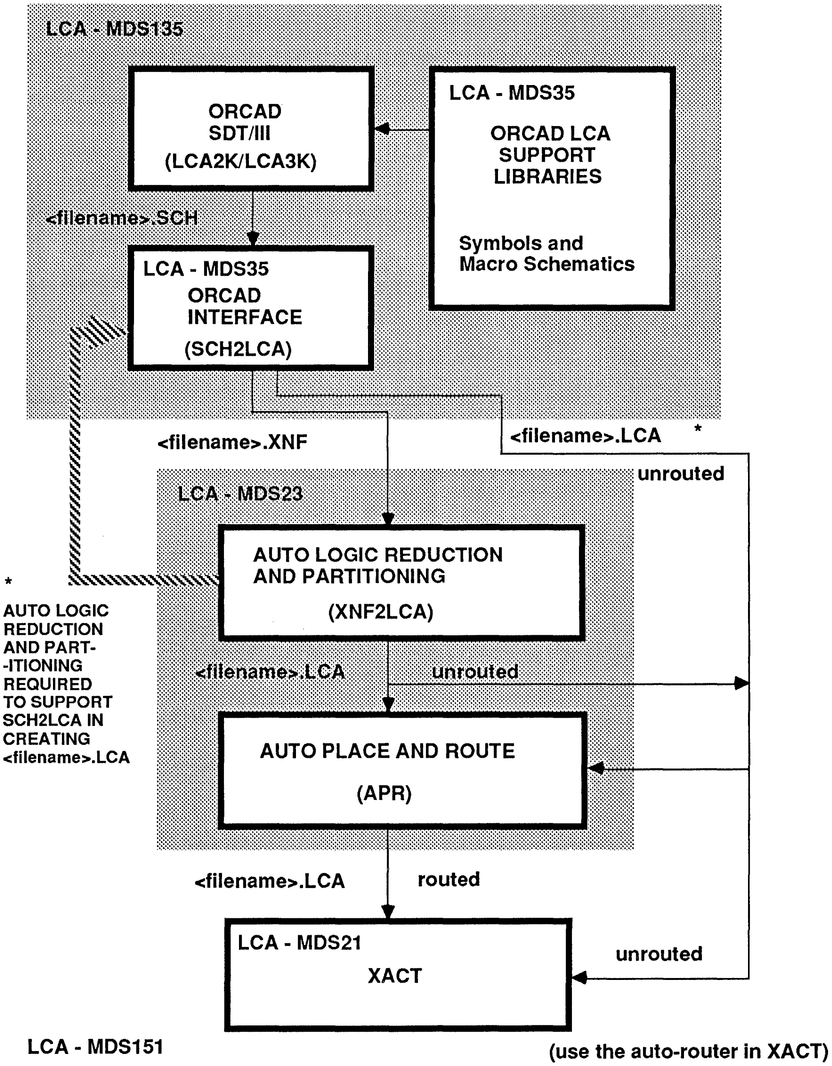


Figure 1. Design Flow Using the OrCAD Schematic Entry System

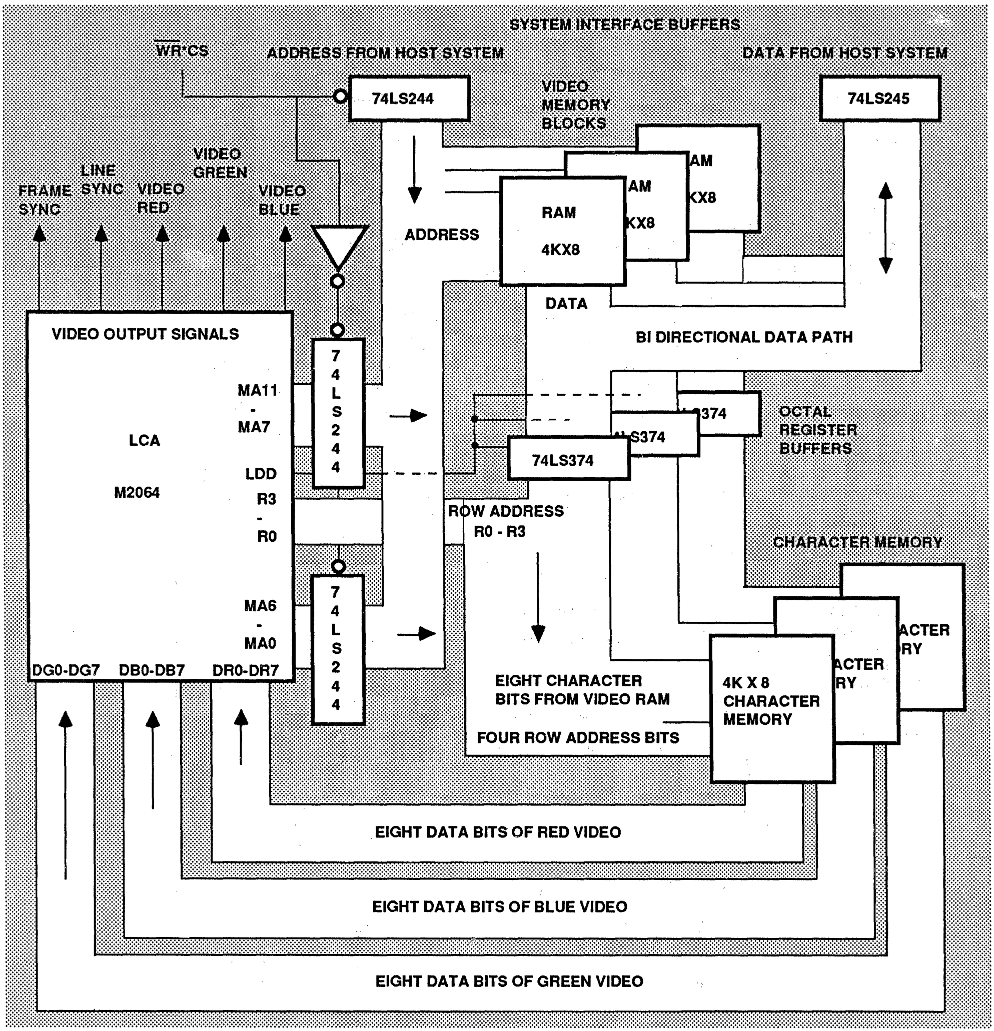


Figure 2. Block Diagram of a Video Controller System

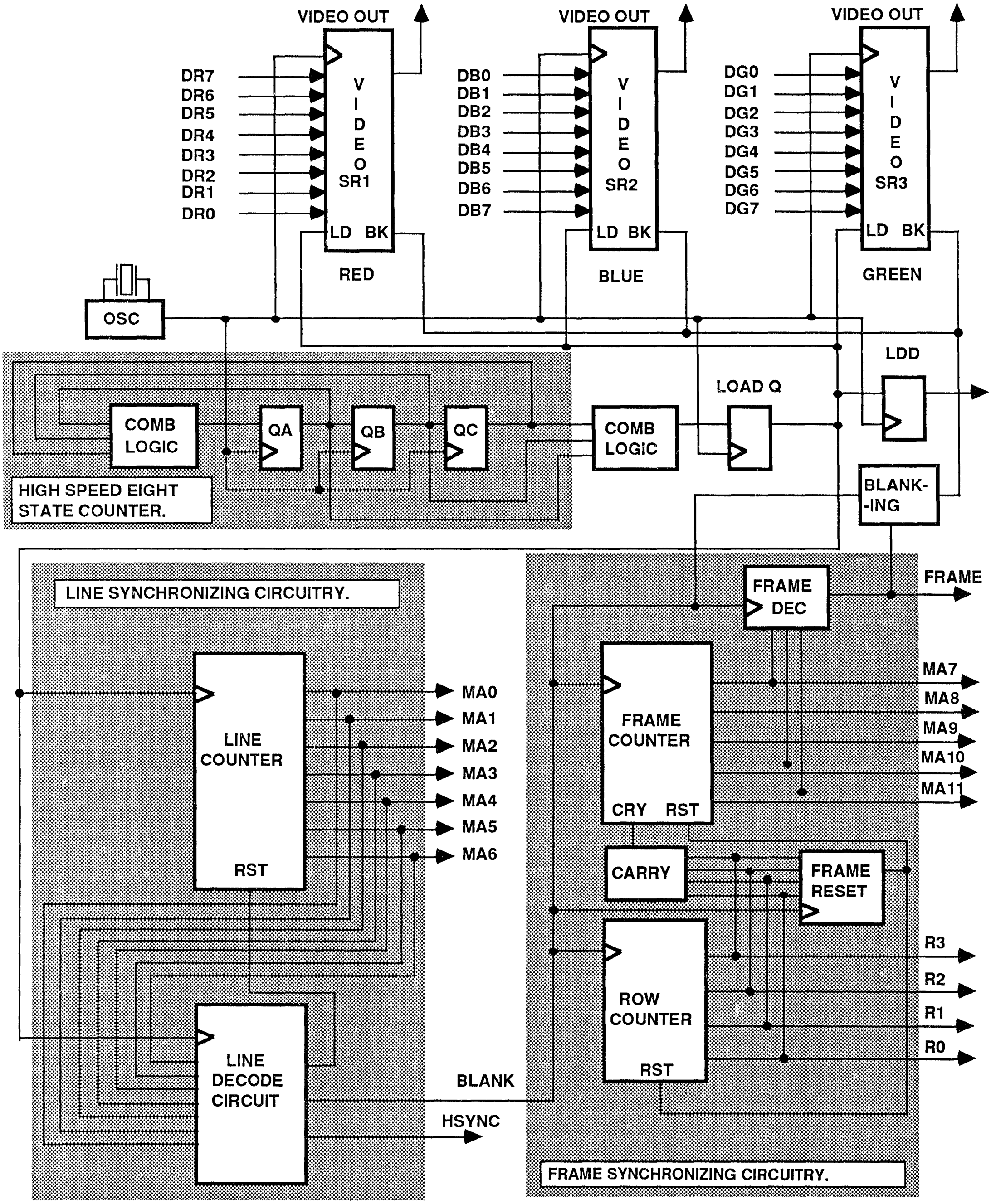


Figure 3. Block Diagram of the Video Controller Design

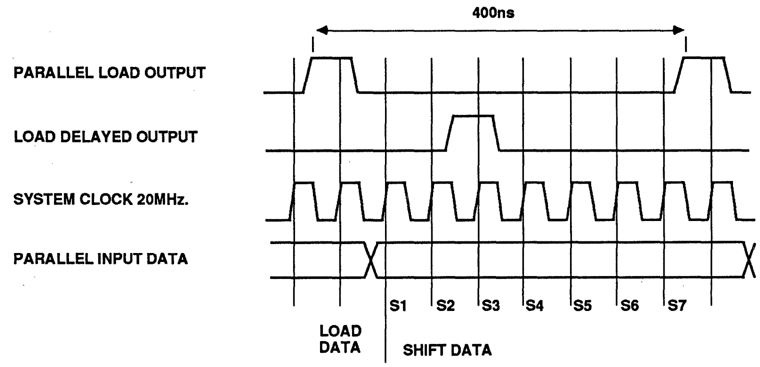


FIGURE 4a.

One parallel load and seven shift states are completed for one complete row of character data. The rising edge of the parallel load signal clocks the memory address counters MA0 - MA6.

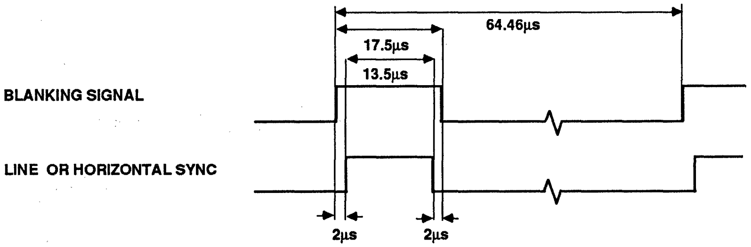


FIGURE 4b.

The line sync, or horizontal sync pulse has a period of 64.5μs. The active blanking pulse envelopes the line sync pulse to give a front and back porch of 2μs

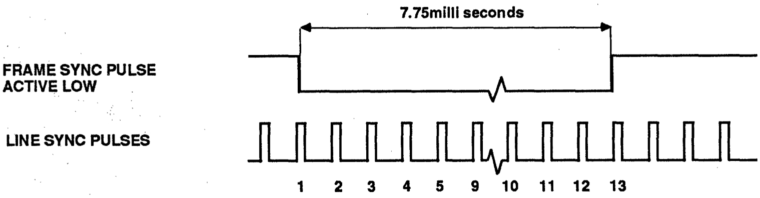


FIGURE 4c.

The frame sync pulse is asserted by the rising edge of the line sync pulse and is held asserted for twelve active line periods. The frame sync pulse goes inactive after the rising edge of the thirteenth line sync pulse.

Figure 4. Video Applications

the next line of display. The horizontal synchronization and blanking pulses are decoded from these outputs and are used internally to clock the row and frame counters and blank the shift registers during horizontal retrace. The timing relevent to these signals is shown in Figure 4b. The row counter outputs R0–R3 access the current row of the character to be displayed in the character memory, which could be a ROM or RAM containing a character font. The end of an active line scan is marked by the rising edge of the blanking pulse, which is used to increment the row address counters. The higher order address lines MA7–MA11 are also incremented by the blanking pulse and these ouputs access the video character to be decoded and displayed. The contents of the RAM that are registered into the 74LS374 buffers point to a base address in the character memory. The row address outputs select the current row indexed from this base address and the data contents are loaded into the shift registers in the LCA device.

The host system can access and update the contents of the video RAMs through interface buffers such as 74LS244 and 74LS245. Also the host system can read a retrace status register in the LCA device so video updates need only take place during line and frame blanking periods.

The outputs MA7–MA11 are decoded to produce a frame synchronization pulse. This signal is normally active LOW but an active HIGH signal is also provided in addition. The duration of the frame synchronization pulse is about 7.75 milli-seconds and lasts for 12 line synchronization periods, Figure 4c shows the timing associated with frame flyback. All the address outputs are reset after a frame synchronization pulse and a new active frame may commence after 13 line sync pulses. Table 1 shows the characteristics of two designs XDES06.LCA and XDES11.LCA as developed for an M2064-70 LCA device.

TABLE 1.

The performance characteristics of the video controller designs have been sumerized below. Two designs were created as applications examples. However, because the LCA device is programmable these formats may be modified for the user's more specific application. The initalization data can be stored in an EPROM, and a number of separate configuration patterns for the LCA may be contained in one EPROM. The required configuration pattern may be chosen prior to the application of power to the device so the video controller may programmed with different display formats.

XDES06.LCA

1.	**Frame frequency**	**51Hz.**
2.	**Line frequency**	**15.5 KHz.**
3.	**Visible Characters per line**	**128.**
4.	**Characters lines per frame**	**24.**
5.	**Dots per character**	**8.**
6.	**Lines per character**	**12.**
7.	**Visible Lines per frame**	**288.**
8.	**Dot clock**	**20MHz.**
9.	**Frame blanking cycle**	**7.8ms.**
10.	**Line blanking cycle**	**13.8µs**

XDES11.LCA

1.	**Frame frequency**	**55 Hz.**
2.	**Line frequency**	**22 KHz.**
3.	**Visible Characters per line**	**80**
4.	**Character lines per frame**	**24.**
5.	**Dots per character**	**8.**
6.	**Lines per character**	**16.**
7.	**Visible lines per frame**	**384.**
8.	**Dot clock**	**20MHz**
9.	**Frame blanking cycle**	**8.8ms.**
10.	**Line blanking period**	**13.5µs**

The Design in OrCAD Schematic Capture

OrCAD/SDT III with its XACT compatable support libraries was used to create the design as a logic schematic. To invoke the program the designer enters the directory ORCAD\LCADSN1, which is the default directory created at the initial installation, and types the command LCA2K <filename>. This schematic package is used to support the design of 2000 series parts. Another software package, LCA3K, is available for the 3000 series of devices. The design was built up from a number of hierarchies starting with the video controller as a block unit in the root hierarchy, shown in Figure 5. Using hierarchy is the most practical way to enter a complex circuit design. The device inputs and outputs are connected to input pads and buffers. The input and output pads are selected from the support library using the GET command in OrCAD. A group of libraries are displayed for selection. The designer should use the IOB library to configure the pad and buffer into the schematic.

To assign a pad to a specific pin the user can label the pad; P27 will fix the input pad labelled DR0 to pin 27 of the LCA device. Input and output buffers are used in conjunction with the pads. These will be configured as buffers in the Input Output Blocks (IOBs) of the LCA device.

The GXTL symbol is retrieved from the support library and the output is connected to the OSC net which is an input to the video controller circuit. The GXTL symbol is consistent with the GXTL macro used in the XACT software and functions as the dot clock oscillator in the circuit. Its frequency of operation is controlled by a crystal source. All events related to loading and shifting of the video shift registers are synchronized to this signal.

Design Hierarchy

The video controller is represented as a block diagram in Figure 5 with inputs and outputs that are labelled inside the boundaries of the block. These net names must be the same as the net names used in the next lower order hierarchy to provide the necessary consistant link through the various sheets in the hierarchy structure. For example, the signal DR0 corresponds with signal DR0 in the next level of hierarchy, as shown in Figure 6.

Figure 6 shows a block diagram of the subsystems that make the video controller. The subsytems are three video shift registers, VIDEORED, VIDEOGRE and VIDEOBLU, a high speed eight state counter, HIGHSPEED8, line and frame synchronization circuitry, and memory addressing circuitry, LINESYNC and FRAMES respectively.

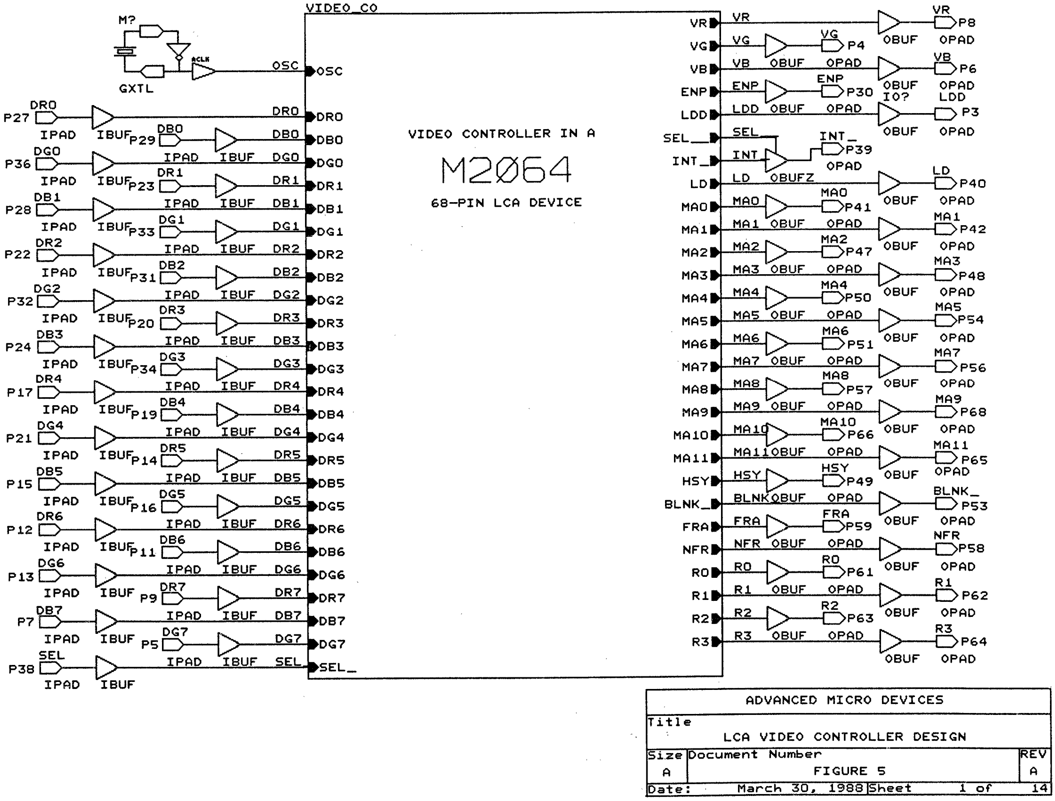


Figure 5. LCA Video Controller Design

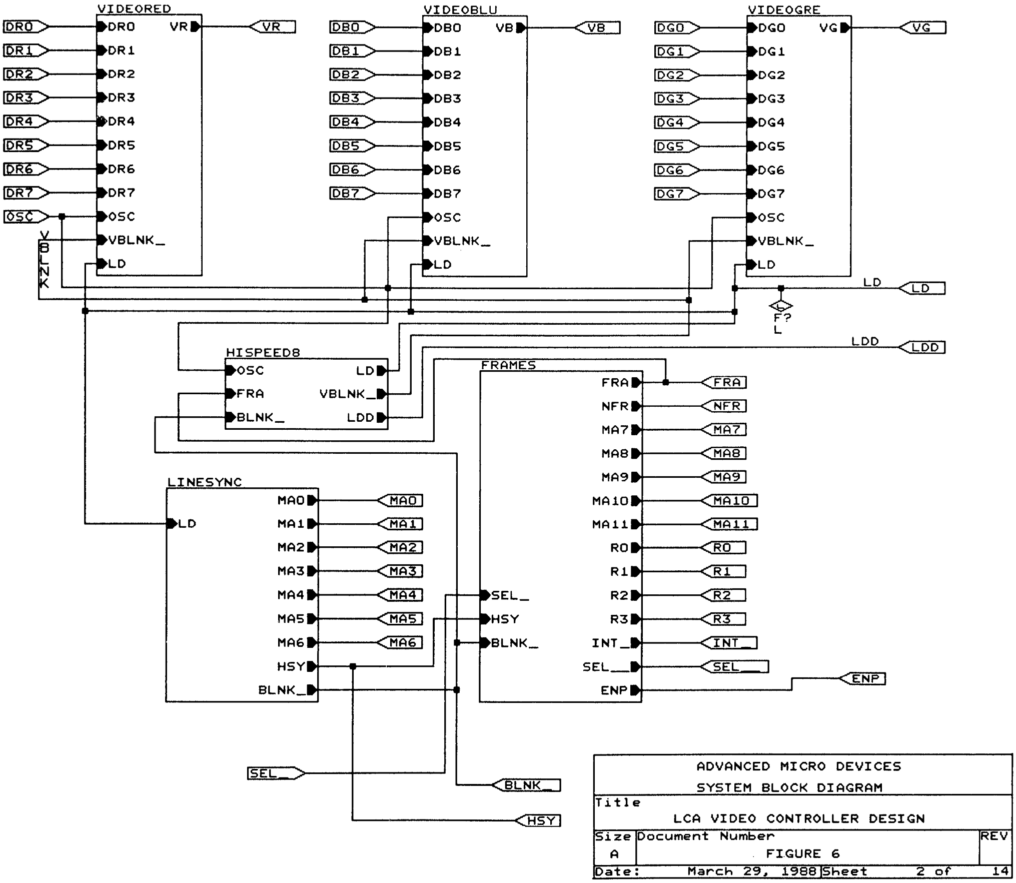


Figure 6. LCA Video Contoller Design

The three shift registers, each identical in design are VIDEORED, VIDEOBLUE and VIDEOGREEN. The eight data inputs DR0-DR7, in the VIDEORED block are loaded into the shift register synchronously by the rising edge at the OSC input. Loading is enabled when CKP_ is HIGH; otherwise, shifting takes place. When VBLNK_ is HIGH, blanking of the three shift registers occurs during both line and frame flyback. The signals in Figure 6, DR0-DR7, OSC, LD, etc., are shown in blocks called module ports which relate the signals to the root sheet shown in Figure 5. In Figure 6, inputs and outputs are shown in the blocks, and are referred to the next lower level of hierarchy. For the shift register VIDEORED the lowest level of hierarchy is shown in Figure 7, where the logic circuit is illustrated as the conventional gates and registers. The module ports OSC, CKP, VBLK, DR0, etc., provide the necessary input from the video shift register VIDEORED in Figure 6. In both Figure 6 and Figure 7 net flag labels have been shown in a diamond-shaped box. For example, the CKP_ line in Figure 7 has a long-line flag attached and in Figure 6 the LD net has a long-line flag. When the circuit is converted into the LCA these nets will be hooked onto long lines.

It is recommended to use longlines whenever signals are required to propagate over the length and/or breadth of the LCA device. Longlines are fixed metal interconnects and should be used when the minimum amount of signal skew is required. Other net information can be superimposed on the design. Some examples of flag usage would be: C for critical (routed first), K for clock input to the CLB and G for a clock input that comes from the combinational G circuitry in the CLB.

In Figure 6 the high speed eight state counter is drawn as a block with inputs OSC, FRA, BLNK_ and outputs CKP_ and VBLNK_, Figure 10 shows the lowest level of hierarchy of this circuit. The three registers form an eight state counter with outputs QA, QB and QC. The 'D' input of the fourth register LOADQ is fed from an AND gate, with three inverted inputs of QA, QB and QC. This circuit provides a logic HIGH output as a LOAD control to the shift registers and other circuitry when the state machine contains a zero count. The 'OR' gate also included in this sheet shows the module port inputs of FRA and BLNK_ gated to provide a blanking pulse for the video shift registers.

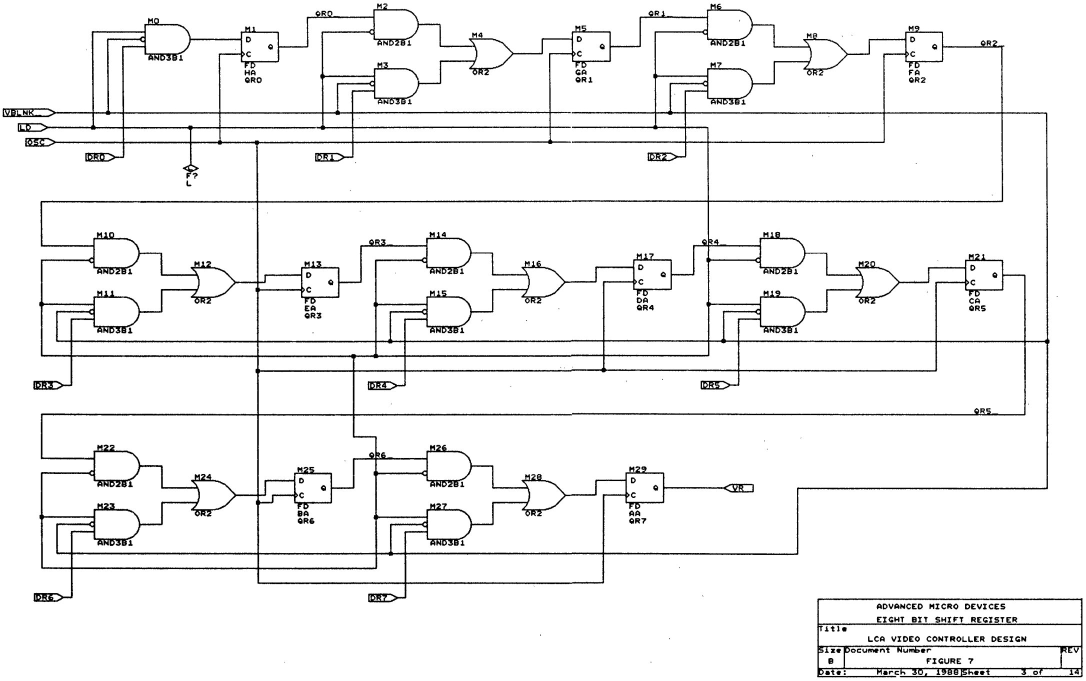


Figure 7. LCA Video Controller Design

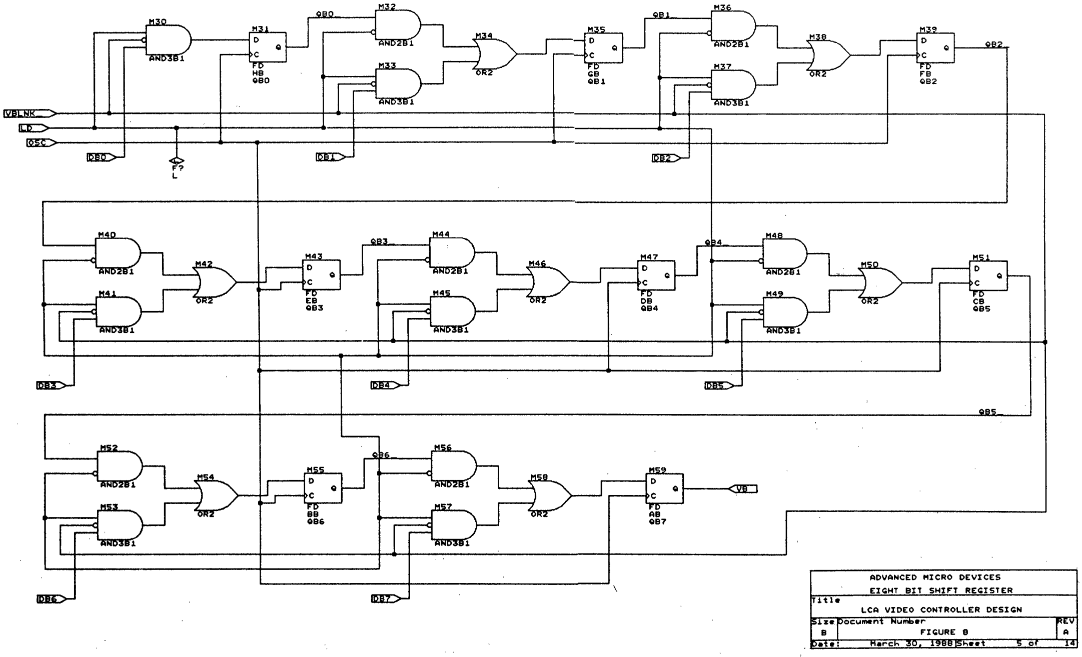


Figure 8

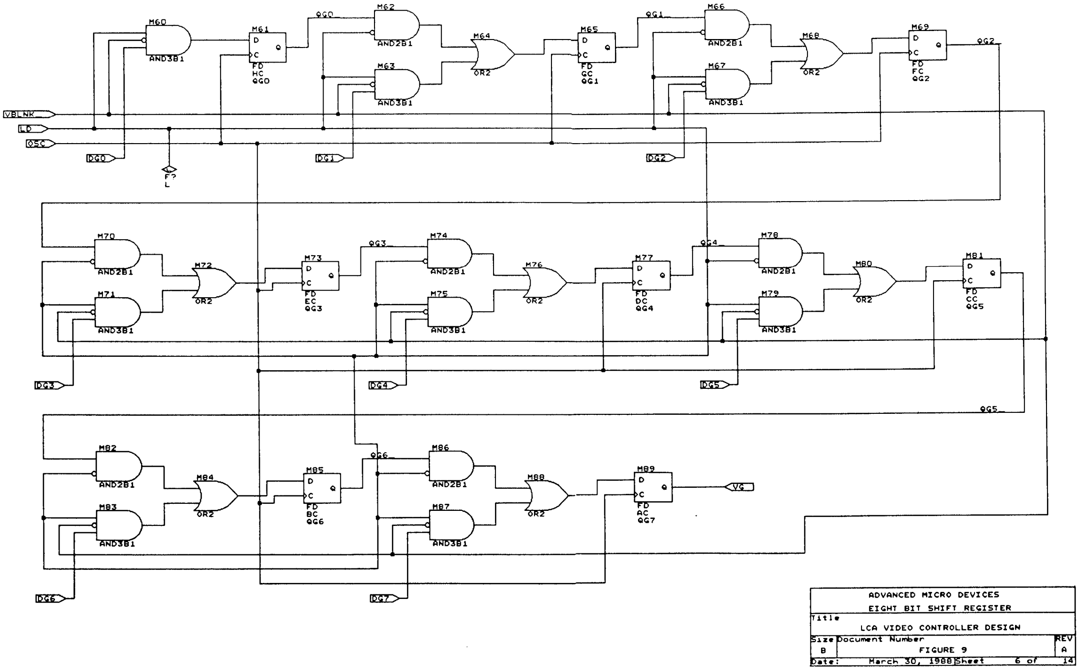


Figure 9. LCA Video Controller Design

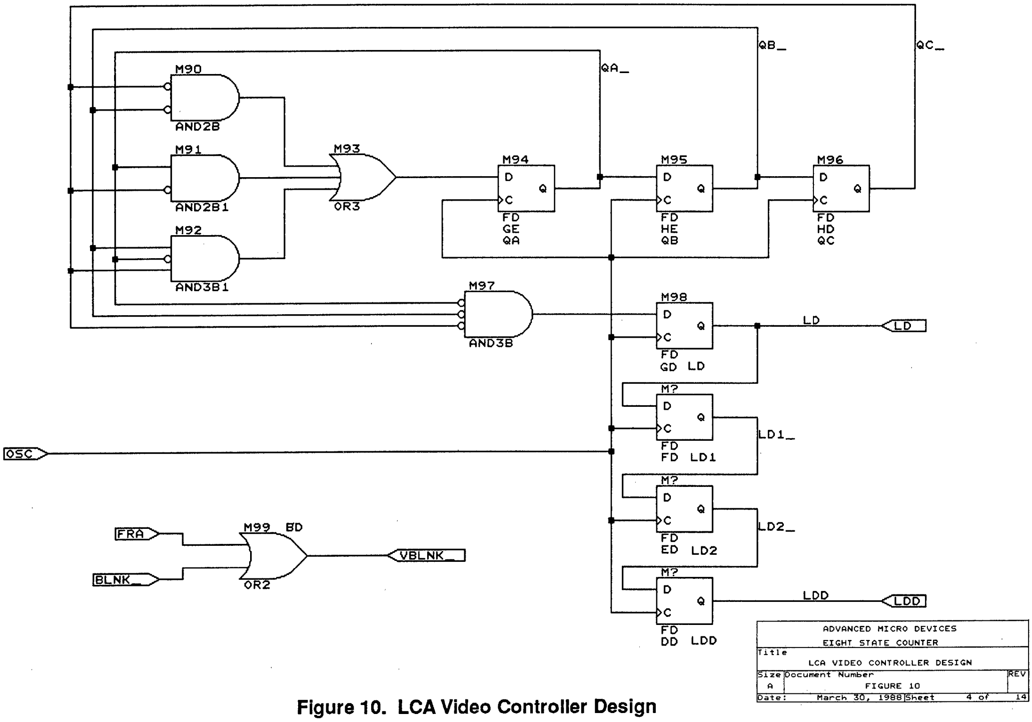


Figure 10. LCA Video Controller Design

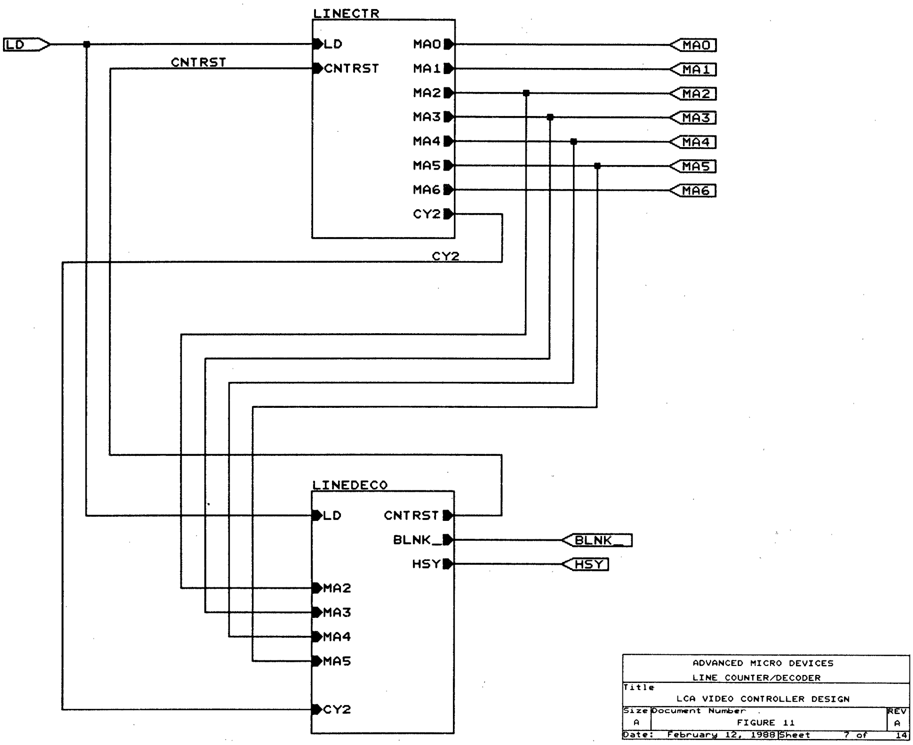


Figure 11. LCA Video Controller Design

Block Placement

Flag constraints which are placed on a design in OrCAD schematic capture are contained in a file generated by OrCAD. The APR program will use the constraints from a file named <filename>.SCP. However it is not always possible to place some logic components to fixed CLB locations in the schematic entry of the design. It may then be necessary to constrain APR to lock some of the logic components to specific block locations. This can be done by the designer, who can enter constraining information into a constraints file <filename>.CST. Appendix 3 shows the constraints file which effectively locked the logic components in the three video shift registers.

OrCAD has provision for identifying gates and registers in the sheet, M90 to M93 in Figure 10 are the reference names of the gates forming the 'sum of product' input to register M94. The circuit is designed as a modified linear shift register to count through eight distinct states. It is a free runing counter from the dot clock oscillator and controls the loading and shifting of video data. Directly beneath the register M94 is the type indication in this case FD is a clocked 'D' type register without SET and RESET inputs. The register can be fixed to a specific location in the LCA device by using the LOCATION command in the EDIT mode. Register M94 has been fixed in the LCA matrix location GE. A name QA has been assigned to the CLB by invoking the BLOCKNAME command. Figure 10 shows the positioning of the other 'D' type registers M95, M96 and M98 at CLB locations HD, GD and HE respectively, along with the block names QB, QC, and LOAD. The separate components of the circuit are connected together using the WIRE command. When converted into the LCA device, default net names are created, or alternatively the designer can assign a net name by using the PLACE then LABEL command. Assigning meaningful names to nets, rather than relying on default labels, can help in the editing of the design at later stages.

Line Synchronization Circuitry

Figure 11 shows the next level of hierarchy down from the LINESYNC block. A line counter and line decoder are designed to provide memory addressing to the character memory and generation of horizontal synchronization and blanking to the Visual Display Unit.

The Line Address Counter is shown in Figure 12, it is essentially a seven bit counter built up from 'D' type registers and discrete gates. The clock input to the Line address counter comes from the LD (load) net, so each time a character is addressed in video memory and loaded into the shift registers the memory address counter is incremented. MA0–MA6 are the counter outputs and CY2 is a carry output to the Line decoder circuit shown in Figure 13. The BLNK_, HSY and CNTRST outputs are derived from the memory address counter in Figure 12. Gating and registers form 'J,K' functions to turn on a blanking control and horizontal synchronization control which initiates line flyback activity.

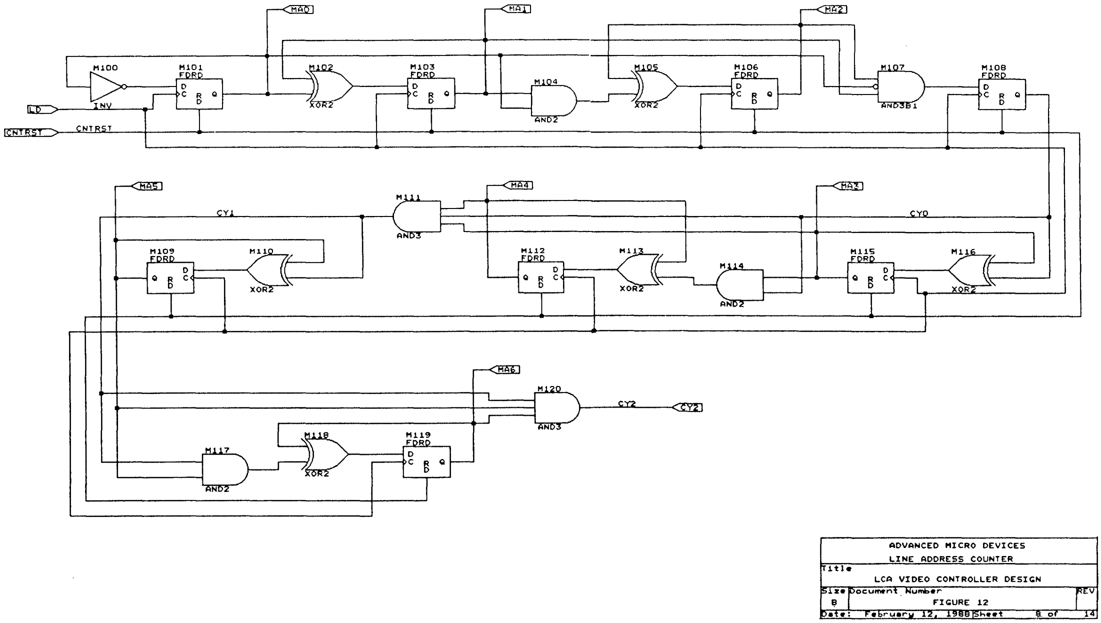


Figure 12. LCA Video Controller Design

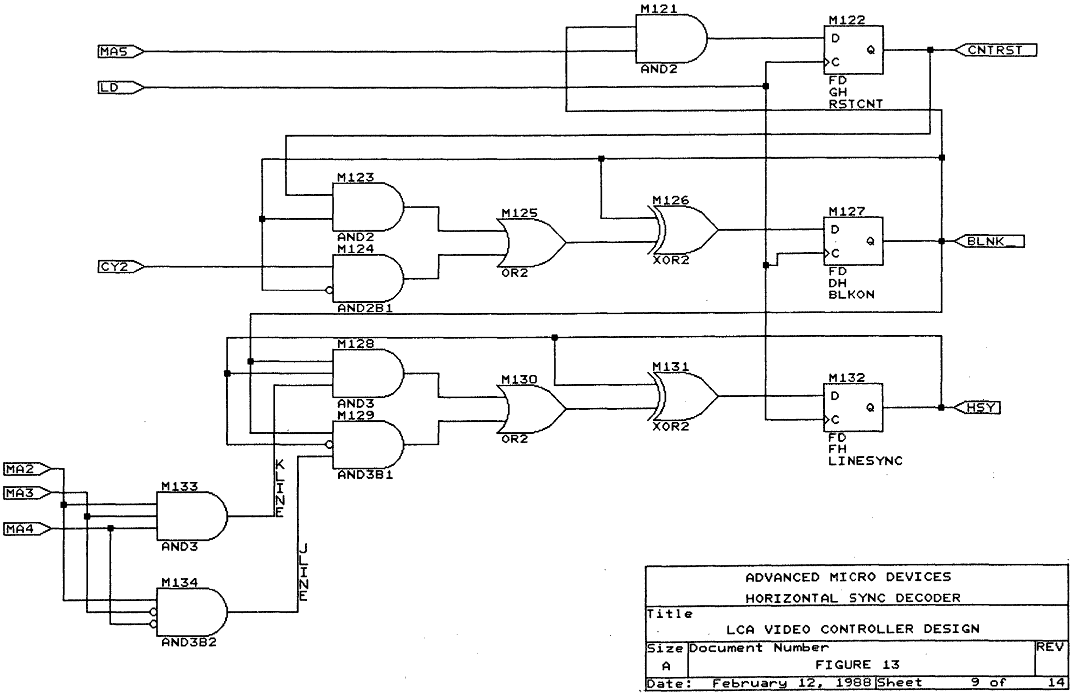


Figure 13. LCA Video Controller Design

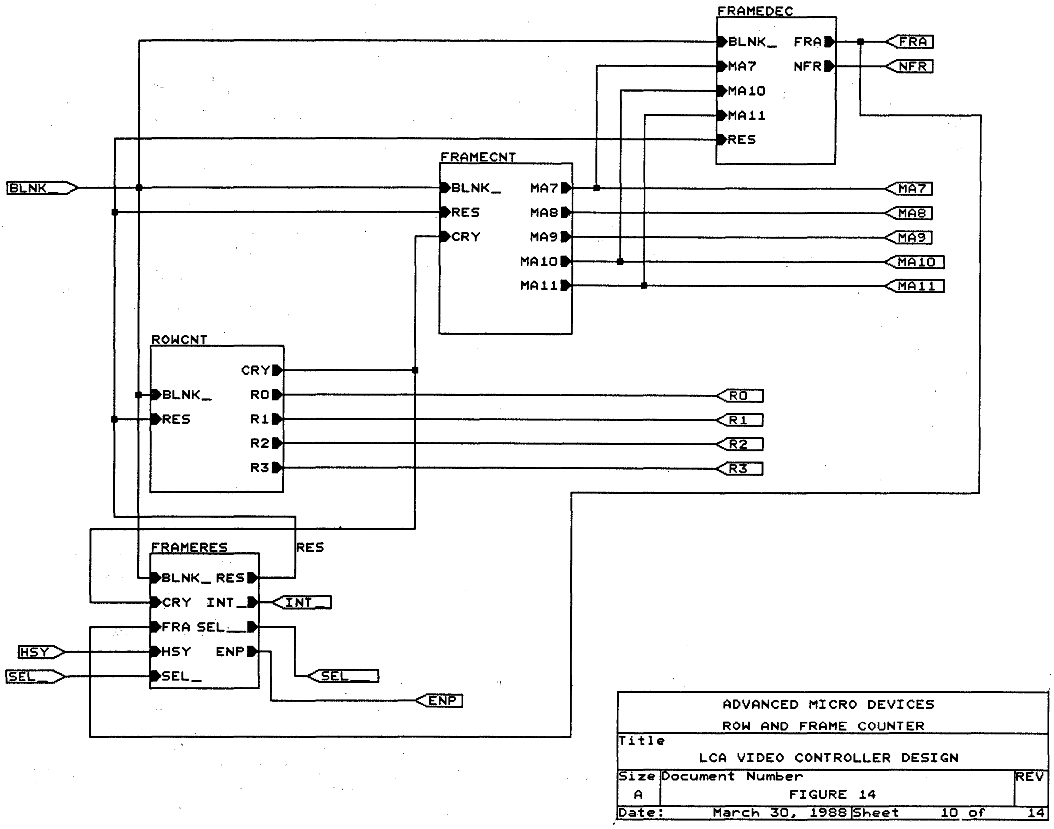


Figure 14. LCA Video Controller Design

Frame Synchronization Circuitry

The FRAMES block in Figure 6 contains a lower level of hierarchy that is shown in Figure 14. The block diagram contains a frame counter FRAMCNT for addressing video memory and frame decoder FRAMEDEC for generating frame synchronization pulses. The ROWCNT block contains a four bit 12-state counter for addressing individual lines of a 12 character row, and the FRAMERES block is designed to reset the counters after one complete frame of information has been displayed.

The row address counter shown in Figure 17 is incremented each time a blanking pulse is applied to the clock input of the registers. This ensures that the next row of video information is addressed after each line scan. The CRY output is connected to the frame counter in Figure 16 enabling it to increment after 12 lines of video information have been displayed.The frame counter is shown in Figure 16, outputs MA7 - MA11 address characters in video memory. At the end of displaying one complete row of characters, the CRY input is asserted and enables the counter to increment in response to the rising edge of the line blanking pulse.

The frame sync generator shown in Figure 15 is a 'J K' register designed to assert the output FRA when MA7–MA11 reach a count of 18 hexadecimal (24 decimal), so a total of 24 rows of characters can be displayed. The frame synchronization pulse is turned off after 12 line flyback periods.

Figure 18 shows the reset circuitry that clears all the counters on the commencement of a new frame. When performing a reset it is good practice to use a register to hold reset active for a defined period of time, in this case one line period. In this way the RES ouput is allowed time to propagate through a number of different interconnect routes to properly reset all the registers that require clearing after the display of a complete frame.

Additional circuitry has been included in this Figure 18. A NAND gate provides a LOW output during line and frame flyback. This signal can be used as an interrupt to a host processor so video memory updating can occur during blanking periods.

The ENP(ENable Prom) output has been designed to be connected to the LCA LDC pin. This pin is normally LOW during and immediately after configuration. It will remain LOW until the first frame pulse clocks a logic HIGH through the register to the ENP output. If this pin is tied to a higher order address line, character data may be stored in the same PROM as the configuration data. At this point the component ceases to be redundant.

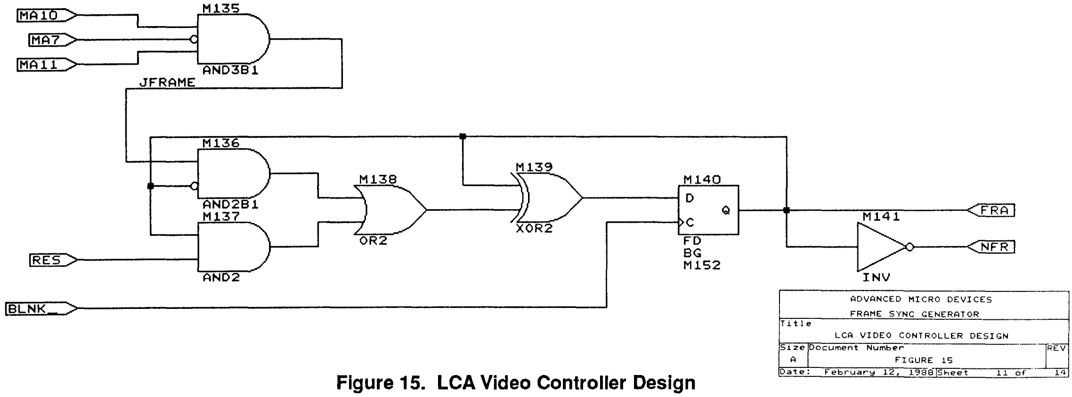


Figure 15. LCA Video Controller Design

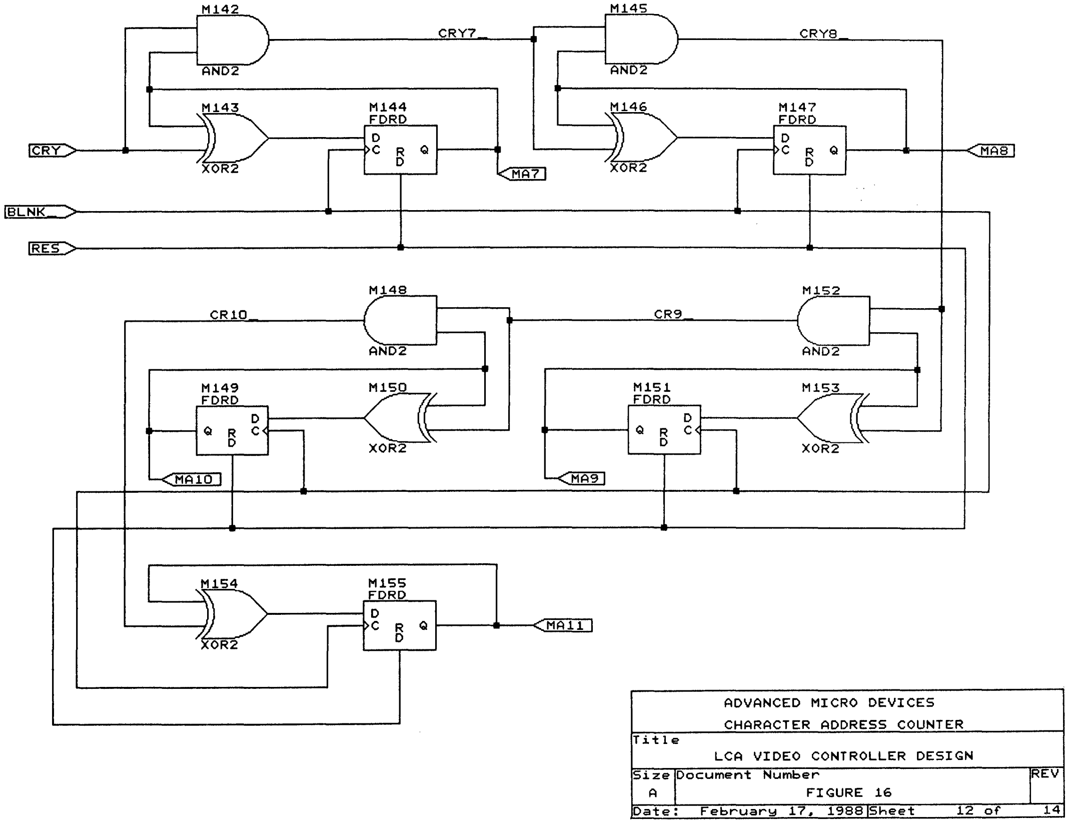


Figure 16. LCA Video Controller Design

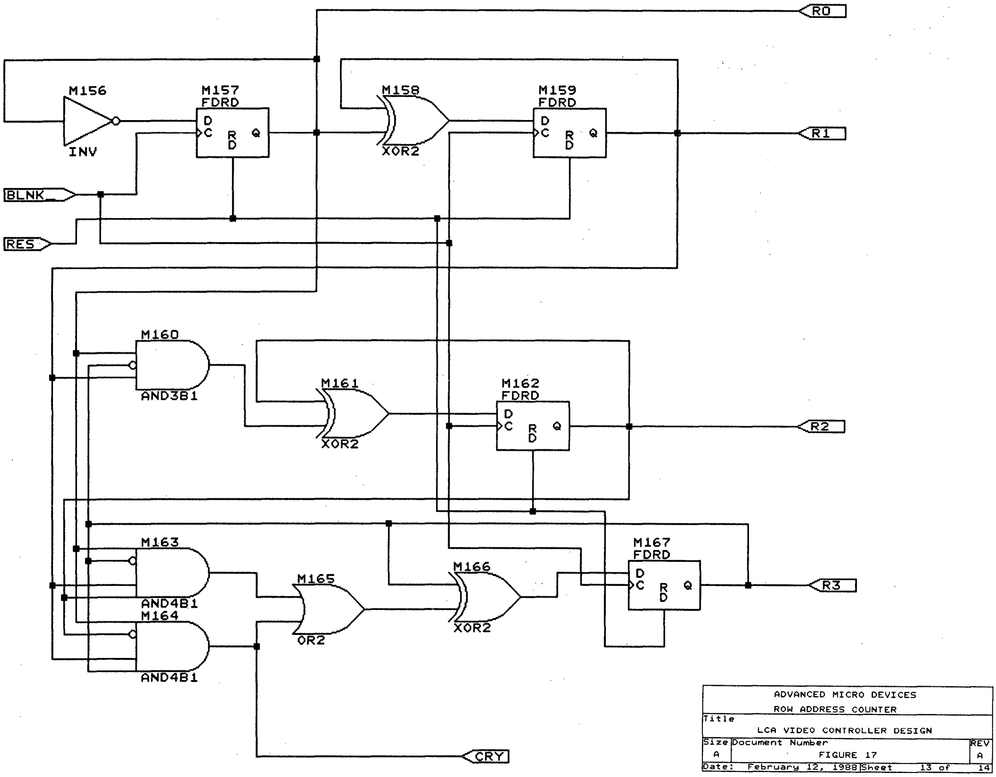


Figure 17. LCA Video Controller Design

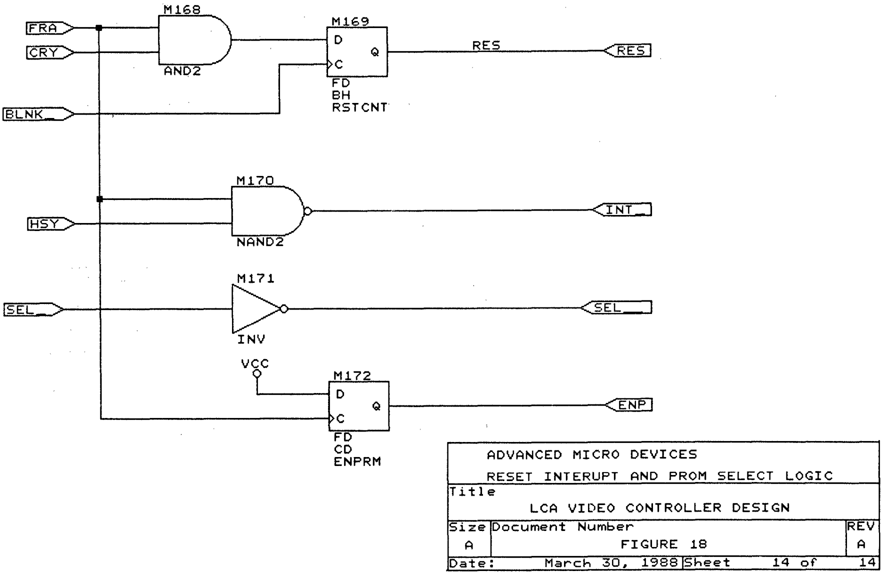


Figure 18. LCA Video Controller Design

Summary

Figure 19 shows the ultimate layout of the design once it is established into the LCA device. The pinout information is shown for ease of wiring into a system. Further logic manipulation can be accomplished in the XACT environment. The ability to store a number of different LCA design configurations in a single EPROM means that a choice of different display formats are possible with the minimum hardware change. Switch selection on the higher order address lines of the EPROM can be used to select one of a number of different configuration patterns. In a 2064 EPROM the storage of eight configuration patterns possible. This gives a high degree of flexibility in using the LCA device as an overall systems design solution.

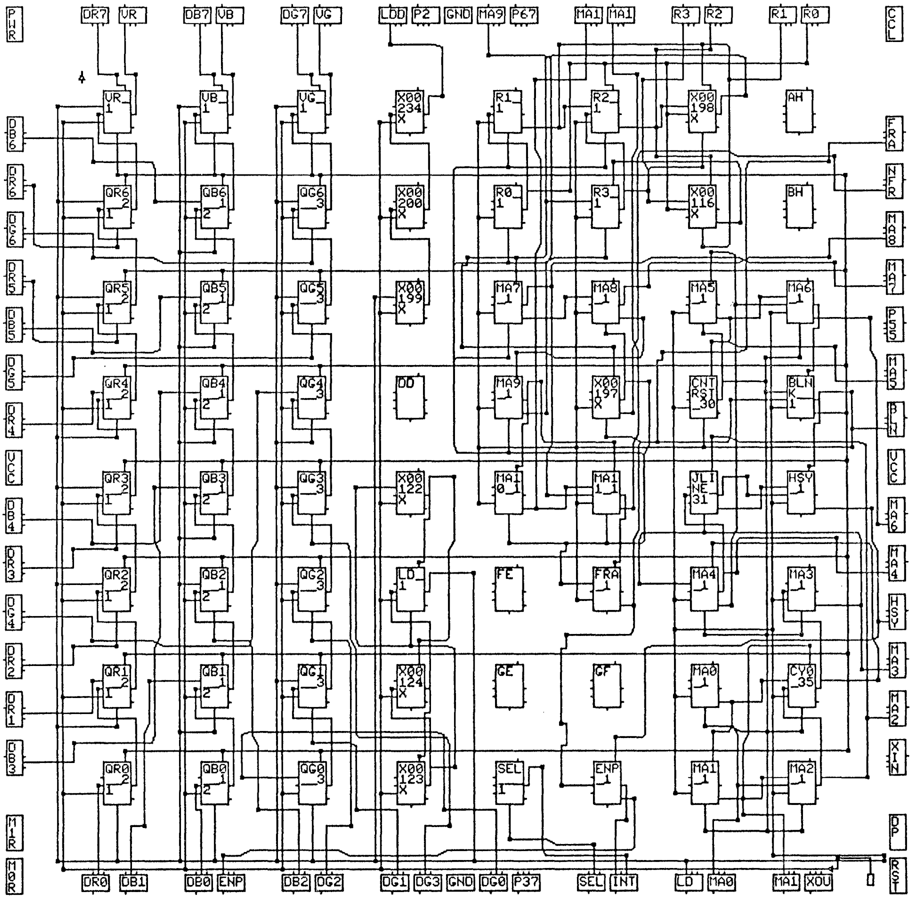


Figure 19. LCA Video Controller Design

APPENDIX 1

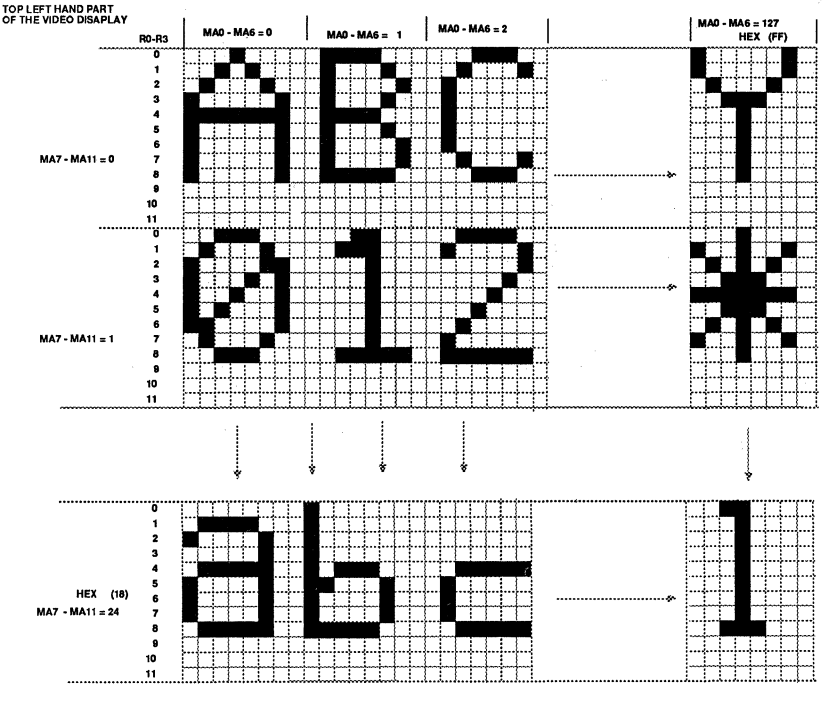
Figure 1 shows a how alphabetic characters are displayed on a video screen. At the top left corner of the screen all the address counters are cleared. The first video character to be displayed is addressed by MA0–MA11, in this case the ASCII code for 'A' is read from video RAM and fed to the higher order address inputs of a character PROM referencing the dot pattern for the letter 'A'. The row address inputs R0–R3 access the first eight-bit pattern to be loaded into the shift registers. One load and seven shift cycles take place. The memory address counter MA0–MA6 is incremented to 1 and the code for 'B' is accessed in the video RAM then fed to the character generator PROM. The row counter output is still zero, so the first row of letter 'B' is accessed and loaded into the video register and clocked out at the rate of the system dot clock. The memory address counter MA0–MA6 increments to 2 to access the code for 'C' from the video RAM and the first row of that character is clocked out in the same way as the previous two characters.

The row counter is not incremented until the complete line has been displayed. The line display is terminated with a blanking and horizontal sync pulse. The blanking pulse is used to clock the row counter from zero to one. The next row of characters 'A', 'B' and 'C' are sequentially accessed for display. MA0–MA6 is set to zero at the begining of each line scan to build up the display as a series of line scans.

In this specific example lines (rows) 0 to 8 form the first complete character row, with lines (rows) 9, 10 and 11 forming a margin before the next row of characters 0, 1, 2...* is accessed and displayed.

The higher order memory address lines MA7–MA11 increment to access the next row of characters, and the line(row) counter is reset to zero.

The whole frame is built up as a series of lines(rows) forming a row of characters which combine to produce a complete frame of character information.



The frame starts at the top left hand corner of the display with R0–R3 = 0, and MA0–MA11 = 0. The first row of the letter 'A' is displayed, by clocking the video shift register seven times. The memory address increments to MA0–MA6 = 1. The top row of the letter 'B' is clocked out of the video shift register. The memory address is incremented to 2 and the top row of the dot pattern of the letter 'C' is accessed, and shifted. The last letter in the complete line of 80/127 characters is 'Y' and the first row of this character is accessed and displayed. A blanking pulse followed by a line sync generation pulse resets the memory address counter MA0–MA6, and the row address counter increments to piont to the next row. The same characters are accessed and the next row of each character is accessed and clocked into the video shift register. After the complete display of one charcter row of twelve lines, the address output MA7–MA11 increments to point to the next set of characters to be displayed. The sequence continues for 24 rows and is terminated by a frame sync pulse.

Figure 1.

APPENDIX 2

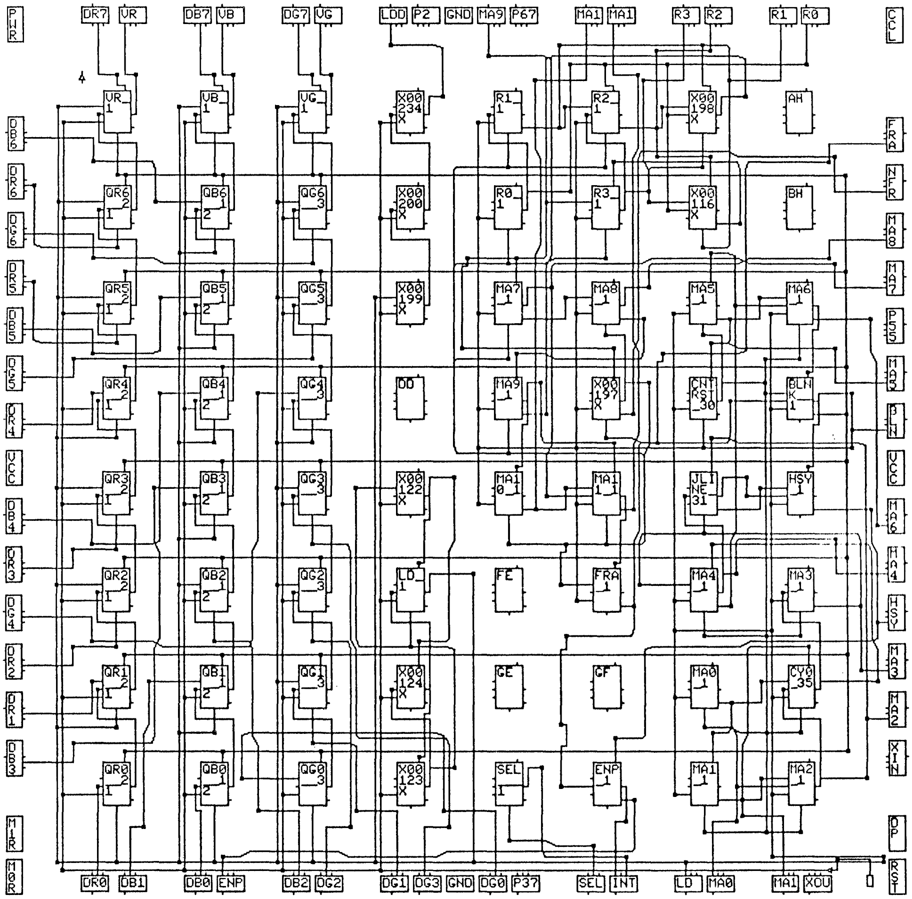


Figure 19. LCA Video Controller Design

APPENDIX 3

The positioning of registers in the three shift register designs has been locked into defined block locations by this constraints file. The output net of the logic circuit is listed next to the matrix grid location of the block. For example the net output VR_1 and its associated logic configuration has been placed by the designer at block location AA.

```
PLACE BLOCK VR_1     AA ;
PLACE BLOCK QR6__21 BA ;
PLACE BLOCK QR5__21 CA ;
PLACE BLOCK QR4__21 DA ;
PLACE BLOCK QR3__21 EA ;
PLACE BLOCK QR2__21 FA ;
PLACE BLOCK QR1__21 GA ;
PLACE BLOCK QR0__21 HA ;

PLACE BLOCK VB_1     AB ;
PLACE BLOCK QB6__12 BB ;
PLACE BLOCK QB5__12 CB ;
PLACE BLOCK QB4__12 DB ;
PLACE BLOCK QB3__12 EB ;
PLACE BLOCK QB2__12 FB ;
PLACE BLOCK QB1__12 GB ;
PLACE BLOCK QB0__12 HB ;

PLACE BLOCK VG_1    AC ;
PLACE BLOCK QG6__3 BC ;
PLACE BLOCK QG5__3 CC ;
PLACE BLOCK QG4__3 DC ;
PLACE BLOCK QG3__3 EC ;
PLACE BLOCK QG2__3 FC ;
PLACE BLOCK QG1__3 GC ;
PLACE BLOCK QG0__3 HC ;
```

Advanced
Micro
Devices

Configuring the Logic Cell Array through the 8051 Microcontroller Family

Chris Jay, Thy-Hien Le, Darryl Engel and Bill Hollon

Publication #	*Rev.*	*Amendment*	*Issue Date*
12024	**A**	**/0**	**8/89**

Configuring the Logic Cell™ Array through the 8051 Microcontroller Family

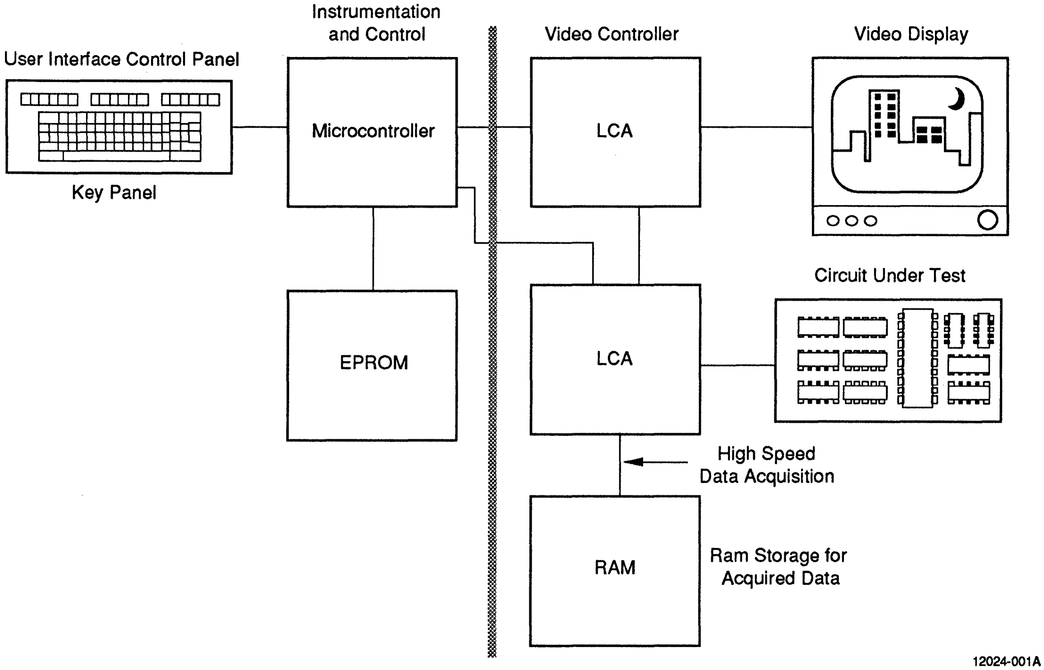
by Chris Jay, Thy-Hien Le, Darryl Engel and Bill Hollon

INTRODUCTION

The Am8051 is the central core of an entire family of 8-bit single-chip microcontrollers that have the processing capability of microprocessors with the added features of on-chip timers/counters, UARTs, RAM and, in some cases, ROM or EPROM. In addition, they have programmable I/O ports for communication to and from systems and devices under their control. Later products such as the Am80535 and Am80515 have an increased number of ports, timers etc., plus analog-to-digital converters for 'real-world' interface. These programmable devices can be used for a wide range of controlling functions: vending-machine control, car-trip computers, washing-machine control, cash-point terminals, cash registers, microwave ovens. The list is virtually endless. In many cases it may be necessary to have support logic circuitry to provide a complete system solution. If this is the case the Logic Cell™ Array (LCA™) is an ideal partner for the microcontroller because it can perform logic tasks that would otherwise overburden the microcontroller; or, it could take on high-speed logic functions that the microcontroller is unable to perform.

The LCA is a dynamically programmable device that must go through a configuration cycle each time power is initially applied to the circuit. There are advantages in this feature. If the designer requires the LCA to perform alternative logic functions, for operation in different environments, then provision can be made for loading the most appropriate configuration pattern. Moreover, any printed circuit card containing a microcontroller and an LCA is totally uncommitted to any specific function. Both the controller and LCA may be programmed at a later time to fulfill one of many different controlling operations. The printed-circuit layout will be less prone to error and have fewer interconnections because most of the pc layout will migrate into the LCA as internal interconnect, and into the firmware of the microcontroller. The overall result is a totally flexible approach to controller applications.

HIGH SPEED STATE MACHINE LOGIC ANALYZER


Instrumentation and Control
Video Controller
Video Display
User Interface Control Panel
Microcontroller
LCA
Key Panel
Circuit Under Test
EPROM
LCA
High Speed Data Acquisition
RAM
Ram Storage for Acquired Data

12024-001A

Figure 1. Microcontroller/LCA Applications

MICROCONTROLLER AND LCA APPLICATIONS

An automatic cash-point terminal in a bank is an example of the two devices working together. The microcontroller is programmed to interface with the bank's computer and control the keypad through which the customers can enter data. Information such as commands or instructions is fed to a video display unit that is controlled by an LCA device. The microcontroller cannot perform this function because it is not suitable for high-speed state-machine design. If the dot clock of the video controller is 18 to 20 MHz, loading and shifting of video information is controlled by a state machine operating at this frequency. The result of this microcontroller/LCA combination is a reduction in the total chip count of the system, since the LCA device can be programmed to absorb a variety of logic functions that would typically require 20 to 40 discrete MSI CMOS devices, depending upon complexity and function.

The logic-analyser circuit, shown in Figure 1 is similar to the example of the cash point terminal. The microcontroller and two 2000-series LCA devices are used to create the essential building blocks. One LCA is programmed to control the visual display; the second is programmed to perform data acquisition of the digital signals under observation.Although two LCA devices are shown, the logic function may be condensed into one of the higher density 3000-series devices. On the acquisition cycle, logic conditions may be sampled and loaded into the RAM. After acquisition, the RAM contents may be read for display on the video display. The role of the microcontroller is to first configure the LCAs, then to interface their functions to the front panel controls. The operator can enter commands via keypads and switches that are interpreted by the microcontroller and passed on to the LCA devices. These commands may result in digital switching levels controlling the function of the LCAs or complete reconfiguration of one or both of the devices. For example, range switching or scrolling of the displayed output could be achieved under logic control, whereas switching between display formats such as waveform display format to a hexadecimal table format might be achieved by complete reconfiguration of the LCA devices.

CONFIGURATION OF THE LCA FROM THE MICROCONTROLLER

The LCA consists of an array of CMOS RAM cells that form Configurable Logic Blocks (CLB), Input/Output Blocks (IOB) and programmable interconnections. Each time power is removed from the system, the LCA loses its configuration data as the CMOS RAM cells discharge. In fact, the device behaves like a conventional static RAM in that respect. Data in RAM, such as look-up tables and reference data, must be set up by the microcontroller after power is appliked. In the same way, configuration information must be downloaded to the LCA device to set it up for logic operation.

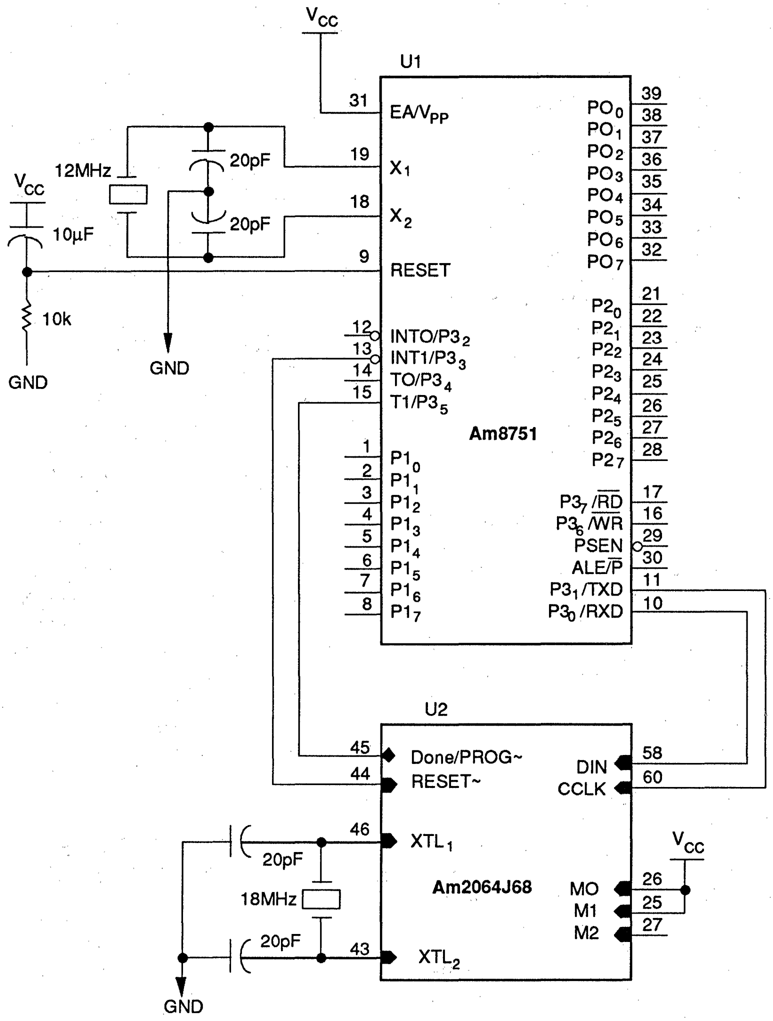
As with microprocessors, microcontrollers need assembly-language code residing in ROM or EPROM. The configuration bit patterns for the LCA may be stored in the same non-volatile memory. On system initalization, this information may be written to the LCA by the host microcontroller on a bit-by-bit basis. Since it has access to the program memory, the microcontroller can read the configuration file one byte at a time, serialize the data, and clock it into the LCA device from a dedicated clock line. This takes two bit positions from one I/O port, which are wired to DIN and CCLK on the LCA. Another port bit position is used to monitor the DONE/$\overline{\text{PROG}}$ pin which is set Low by the microcontroller, and stays Low during the configuration cycle. This line is monitored by the microcontroller so the software can determine when the configuration cycle is completed. The $\overline{\text{RESET}}$ line is taken Low before configuration to reset the configuration circuitry and, on its rising edge, it samples the configuration mode inputs, M0, M1, and M2. After configuration, the microcontroller can take this line Low to reset all the registers in the device. The LCA device can be initialized and reconfigured provided that, after asserting the DONE/$\overline{\text{PROG}}$ and $\overline{\text{RESET}}$ pins Low, a delay of about 160 to 197 cycles (depending on the device used) of the internal sampling clock is given to clear the internal memory during the initialization phase of the configuration.

CONFIGURING THE LCA FROM THE AM8751 MICROCONTROLLER

Probably the simplest circuit solution to an LCA/ microcontroller combination is shown in Figure 2: only two devices are used in this circuit. The Am8751 device is a 40-pin microcontroller with 4K of on-chip EPROM, 128 bytes of internal RAM, two timers and one serial port. As illustrated in Figure 2, to access the EPROM, the $\overline{\text{EA}}$/VPP input pin must be tied High. The Am2064 LCA device requires a little less than 2 Kbytes of configuration data so about half of the fixed EPROM memory is left for controller software. Table 1 shows the memory space requirements for three LCA products, and the remaining memory left over for program storage.

Table 1. EPROM Locations

LCA Device	Required	Available
Am2064	05E2 HEX	0A1D HEX
Am2018	08BC HEX	0734 HEX
Am3020	073D HEX	08C2 HEX



12024-002A

Figure 2. Am8751/LCA Application

There are a number of different modes that may be used to configure the LCA device. When working with a microcontroller, the slave mode is the most appropriate, because it requires the fewest pins for configuration. An added advantage is that the serial port can be used for the transmission of digital data. Configuration bits are presented to the LCA DIN line in serial from the RXD (serial input Port P3.0) pin of the microcontroller and clocked by a clock edge appearing at the TXD(P3.1) output. To set this mode of operation, the LCA mode pins M0 and M1 are pulled High and M2 is allowed to float High via an internal pullup.

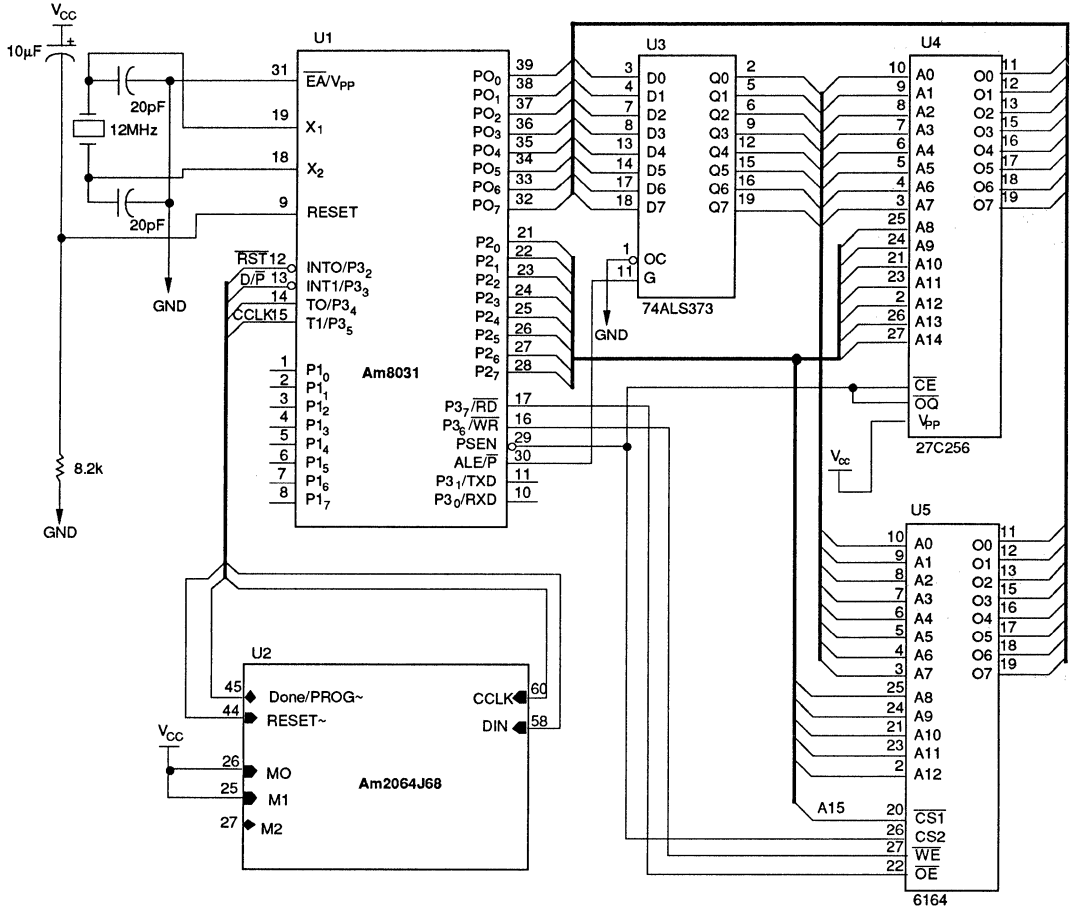
As shown in Figure 2, only four pins of the Am8751 are used for configuring the LCA device: P3.3 is used to pulse the $\overline{\text{RESET}}$ pin Low; P3.5 is used to control the DONE/$\overline{\text{PROG}}$ pin; the serial ports P3.0 and P3.1 are used to send the configuration data and clock pulse via pins DIN and CCLK, respectively. Using four pins of port 3 leaves the remaining four pins of this port and the other three ports of the microcontroller, P0, P1, and P2 available for I/O functions. Pins DONE/$\overline{\text{PROG}}$, $\overline{\text{RESET}}$ and CCLK are dedicated pins; however, the DIN input to the LCA becomes an IOB after configuration, so it may also be used when the LCA becomes a functional device.

After configuration, data may be set up on the RXD (P3.0) output and clocked with the rising edge of the signal coming from the TXD (P3.1) output without affecting the LCA device. So the serial port may be programmed as a UART for general serial communication. Assembly Program Listing gives the instructions, with comments, on how the program was downloaded.

CONFIGURING THE LCA FROM THE AM8031 CONTROLLER

The Am8031 microcontroller is compatible with the Am8751 except that it has no on-chip ROM or EPROM. A separate EPROM is used to store assembly language programs and configuration patterns. Although the Am8031 has 128 bytes of internal RAM, external RAM, can be added to the system. Figure 3 shows an Am8031 controller with a 32K x 8 CMOS EPROM and an 8K X 8 dynamic RAM. The octal latch 74LS373, U3, is used to hold the address A0–A7 information, because the microcontroller port lines P0.0–P0.7 are used for address and data in a time division multiplexed fashion. When the Am8031Address Latch Enable (ALE) output goes Low, the current value of the lower address byte is latched into U3; data read/write operations may then take place over P0.0–P0.7. Port 2 contains the higher order address lines A8–A15. These are used exclusively to access external memory for code and data. To operate the microcontroller in this way, the $\overline{\text{EA}}$/VPP input must be tied Low.

Although the chip count of this small system has increased over the Am8751 application, there is an advantage in this design. External EPROM provides more



12024-003A

Figure 3. Am8031 Application

storage space for code, therefore, a larger number of configuration patterns plus executable code can be programmed into the smaller system with the external EPROM than can be programmed into the Am8751. The disadvantage is the loss of two ports (P0 and P2) which now function as memory addressing and data conveyance. However, the LCA may be configured as a port expander if more I/O functions are required.

Figure 4 shows a block diagram of the LCA configured as a memory-mapped peripheral that can be connected to the microcontroller address and data bus. It can be memory mapped so that active data on the bus may be routed to one of four ports: A, B, C or D. Ports A and B are bidirectional; therefore, data can be read through these byte-wide inputs. The two data-direction registers are loaded through data lines D0 and D1. When the address inputs A0, A1 and A2 are High and the device is selected through the CS input, direction information is loaded into the device. Ports A and B differ in that the A inputs are registered and the B inputs are direct. However, it is relatively simple to adjust the configuration pattern for registered or direct inputs on both ports. By entering the XACT™ design editor, the port IOBs may be reconfigured to the most appropriate configuration.

Other features of this design include handshake signals ARDY and ASTB, BRDY and BSTB plus CRDY and DRDY signals for the C and D ports respectively. Address lines A0–A2 provide address selection for ports A,B,C and D. Port A is selected by address bit 0, port B by bit 1, port C by bit 2 and port D by bit 3, while the $\overline{CS}$ input is driven active Low.

Interrupts are available from the A and B ports. When the strobe signals are used to indicate valid data on ports A and B, the relevant interrupt goes active High and is cancelled after data is read from the selected port.

CONFIGURATION USING PORT BITS

A different approach to configuration was taken in this design. Assuming that the application needed exclusive use of the serial port as a UART, the configuration pins of the LCA were connected to port pins P3.2, P3.3, P3,4 and P3.5 for $\overline{RESET}$, DONE/$\overline{PROG}$, DIN and CCLK inputs, respectively. Port 1 is left entirely for controller I/O operations but, as before, DIN becomes an IOB after configuration, so port pin P3.4 can be used with the LCA for functional purposes.

Assembly Program Listing 2 can be used with the logic configuration shown in Figure 3. The LD_LCA subroutine takes configuration data, one byte at a time, into the accumulator, and serializes that data through the carry register. If the current bit in the carry register is High, the DIN line is set High and the clock line is pulsed Low then High. If the carry register content is Low, the DIN line is left Low and the CCLK line is pulsed through port bit P3.5.

MULTIPLE CONFIGURATION PATTERNS

Figure 5 is an extension of the design shown in Figure 3 incorporating eight LCA devices loaded in parallel through port P1. Note that there is no external RAM. The configuration patterns of the eight devices are stored in the EPROM and accessed together. Assembly Program Listing 3 shows the program that takes a byte of each configuration pattern and bit manipulates the eight bytes so that one configuration bit for each LCA is set up prior to it being clocked. Loading the bit patterns in parallel is an alternative method to loading a multiple LCA design in a serial daisy chain. Although this method uses more pins, it takes less time to configure the entire system of LCA devices.

CONVERSION SOFTWARE

In addition to the assembly-language (or program) configure routines, this application offers a conversion software package written in 'C', called MCS2DB.EXE. This package converts the MCS PROM format, generated by XACT, to a raw data-byte format. This format includes configuration information and can be used as an 'include file' at the end of the assembly code and accessed by the microcontroller during the configuration period.

SUMMARY

The combination of a microcontroller and LCA is powerful and flexible. The memory-mapped port expander was chosen for this specific application example. Many other applications are available, limited only by the user's requirements and imagination.

XACT is a trademark of Xilinx.

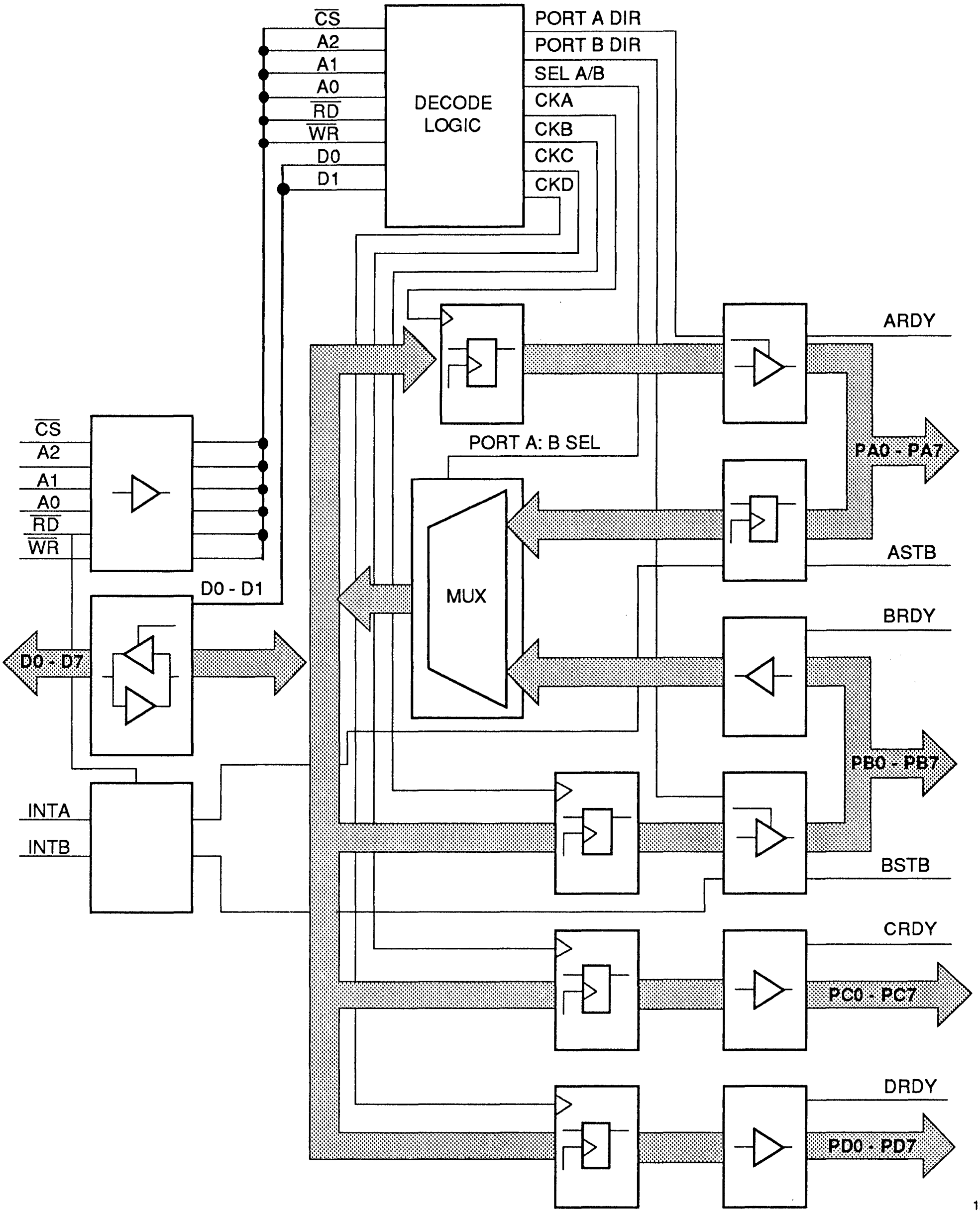


Figure 4. LCA as a Memory-Mapped Peripheral

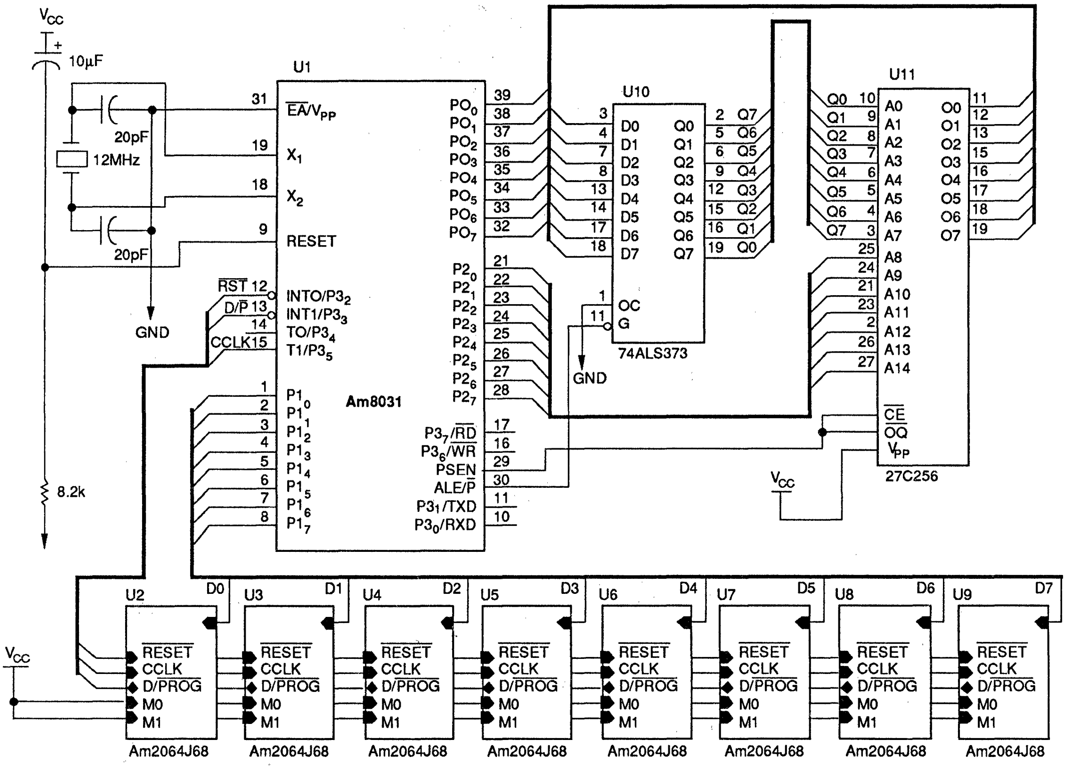


Figure 5. Extended Am8031/LCA Application

12024-005A

```
;       ASSEMBLY PROGRAM LISTING 1.
$MOD51
$TITLE(8751_DOWNLOAD_PROGRAM)
$DATE(21_JULY_1988)
;
;       This program will download the configuration data from
;       the 8751 to the LCA serially.  P3.3 is used as input to
;       the RESET pin.  The DONE pin is monitored through P3.5.
;       The serial ports P3.0 & P3.1 are used to download the
;       configuration data and clock pulse to pin DIN and CCLK,
;       respectively.
;
$MOD51
        ORG     0000H         ;
        LJMP    START         ;
        ORG     0100H         ;Initalize Port3
START:  MOV     P3,#0FFH      ;with all HIGHs.
        CLR     P3.5          ;Drive D/P~ LOW
        CLR     P3.3          ;Pulse Reset LOW
        SETB    P3.3          ;then HIGH
        MOV     P3,#0FFH      ;Release D/~P pin
        ACALL   DELAY         ;Wait for LCA to clear
LOAD:   ACALL   LD_LCA        ;Call subr to load LCA
        JB      P3.5,PGMST    ;Halt if D/~P= 1
        MOV     R1,#0FFH      ;Flag R1
        AJMP    START         ;Load next byte if D/~P = 0
;
;       This subroutine will provide a delay for the LCA to clear
;       the internal memory before another configuration begins
;
DELAY:  MOV     R0,#6FH       ;Load R0 with a number.
IDEL:   DEC     R0            ;Decrement R0 to a delay
        CJNE    R0,#00H,IDEL  ;before doing another load.
        RET
;
;       This subroutine is used to serialy load the LCA
LD_LCA: MOV     P3,#0FFH      ;Set P3 output all HIGH
        MOV     SCON,#00000000B   ;Init the serial port
        MOV     DPTR,#0A00H   ;Load data pointer at
                              ;configuration start.
DATOUT: CLR     A             ;Clear the accumulator
        MOVC    A,@A+DPTR     ;Move data to the ACC
        MOV     SBUF,A        ;Output data serially
        INC     DPTR          ;Increment data pointer
        MOV     A,DPH         ;Check data pointer for
        CJNE    A,#10H,DATOUT ;end of data
        RET
;
PGMST:  MOV     P3,#0FFH      ;Configure P3 as input
;
;       Main program can start here.
;
        ORG     0A00H         ;Configuration data starts here
;
END
```

```
;        ASSEMBLY PROGRAM LISTING 2.
;
$MOD51
$TITLE(8031_CONFIGURATION_PROGRAM)
$DATE(8-AUGUST-1988)
;
;        LCA serial download routine from the 8031 with
;        reconfigurability. The 8031 is configured in the
;        external acess mode. Ports P0 and P2 are wired
;        to EPROM and RAM. The download of data to the
;        LCA DIN pin is through port P3.4. The CCLK pin
;        is connected to port P3.5 with P3.3 and P3.2
;        connected to the D/~P and RESET respectively.
;        Port P1 is available for I/O access. Also, the
;        serial port P3.1 (TXD) and P3.0 (RXD) is available
;        for serial communication.
;
;        Use the program MCS2DB to convert the Intel MCS
;        PROM file to an include file that can be imported
;        to the assembly language program. In this design
;        the origin is at 4000HEX.   Christopher Jay. 8/11/88
;
$MOD51
         ORG      0000H          ;Configuration
         LJMP     START          ;program start at
         ORG      0100H          ;0100H.
;
START:   MOV      P3,#0FFH       ;Configure P1 as I/P.
         CLR      P3.3           ;Drive D/P~ low.
         CLR      P3.2           ;Pulse Reset low then
         SETB     P3.2           ;high. Release D/P~
         SETB     P3.3           ;wait for config
         ACALL    DLAY           ;circuit to clear.
         ACALL    LD_LCA         ;Call subr to load LCA
;
; Wait loop marks end of configuration. Main program
; can start from here.
;
WAIT:    JMP      $              ;Main program can
                                 ;start here.
;
; This subroutine will provide a delay for the LCA to
; clear the internal configuration circuit before
; another configuration can begin.
;
DLAY:    MOV      R0,#7FH        ;Load R0 with vector
IDEL:    DEC      R0             ;7FH Decrement R0 to
         CJNE     R0,#00H,IDEL   ;zero for delay.
         RET                     ;
;
; This subroutine is used to serially load the LCA
;
LD_LCA:  MOV      DPTR,#4000H    ;Initalise data
LOOP1:   CLR      A              ;Clear ACC get
```

Assembly Program Listing 2 (Cont.)

```
         MOVC     A,@A+DPTR      ;config byte.Init
         MOV      R0,#08H        ;loop of eight.
SHIFT:   RRC      A              ;Shift bit into
         CLR      P3.5           ;carry. Clear
         CLR      P3.4           ;CCLK Clear DIN
         JNC      NOBIT          ;test the
         SETB     P3.4           ;carry register.
NOBIT:   SETB     P3.5           ;Set DIN,CCLK.
         DJNZ     R0,SHIFT       ;Shift next bit.
         INC      DPTR           ;Increment data
         MOV      A,DPH          ;Pointer. Test
         CJNE     A,#46H,LOOP1   ;for block of
         RET                     ;2064 config
;                                ;data moved.
;
         ORG      4000H          ;Configuration data
;                                ;starts here @ 4000H.
;
END
```

```
;         ASSEMBLY PROGRAM LISTING 3.
;
;         Primary Controls
$MOD51
$TITLE(MULTCONFIG)
$DATE(19-AUG-88)
;
;         THE PROGRAM HAS BEEN WRITTEN TO
;         CONFIGURE UP TO 8 2064 LCA DEVICES IN
;         PARRALLEL. CONFIGURATION DATA FOR
;         EACH LCA IS STORED IN BLOCKS AND
;         ACCESSED A BYTE AT A TIME. BIT
;         MANIPULATION WILL MERGE BITS FROM
;         EACH OF THE EIGHT BYTES ALLIGNING
;         THEM TO PORT 1 AS P1.0 = DIN(LCA 1)
;         P1.1=DIN(LCA 2) UP TO P1.7=DIN(LCA 0)
;         WHEN ALL BITS ARE ALLIGNED TO THE
;         PORT THE CCLK INPUT TO ALL THE LCA
;         DEVICES IS STROBED. AFTER EIGHT BITS
;         IN THE BYTE HAVE BEEN CLOCKED INTO
;         THE LCA CIRCUITS THE NEXT BYTE IS
;         TAKEN FROM EACH CONFIGURATION PATTERN.
;         THIS IS DONE UNTIL ALL THE BITS
;         HAVE BEEN SENT.
;
CFGAD1    EQU       50H            ;CONFIG ADDRESS 1
LOWAD1    EQU       51H            ;LOW ADDRESS 1
CFGAD2    EQU       52H            ;CONFIG ADDRESS 2
LOWAD2    EQU       53H            ;LOW ADDRESS 2
CFGAD3    EQU       54H            ;CONFIG ADDRESS 3
LOWAD3    EQU       55H            ;LOW ADDRESS 3
CFGAD4    EQU       56H            ;CONFIG ADDRESS 4
LOWAD4    EQU       57H            ;LOW ADDRESS 4
CFGAD5    EQU       58H            ;CONFIG ADDRESS 5
LOWAD5    EQU       59H            ;LOW ADDRESS 5
CFGAD6    EQU       5AH            ;CONFIG ADDRESS 6
LOWAD6    EQU       5BH            ;LOW ADDRESS 6
CFGAD7    EQU       5CH            ;CONFIG ADDRESS 7
LOWAD7    EQU       5DH            ;LOW ADDRESS 7
CFGAD8    EQU       5EH            ;CONFIG ADDRESS 8
LOWAD8    EQU       5FH            ;LOW ADDRESS 8
CFIGD1    EQU       40H            ;CONFIG DATA LOC 1
CFIGD2    EQU       41H            ;CONFIG DATA LOC 2
CFIGD3    EQU       42H            ;CONFIG DATA LOC 3
CFIGD4    EQU       43H            ;CONFIG DATA LOC 4
CFIGD5    EQU       44H            ;CONFIG DATA LOC 5
CFIGD6    EQU       45H            ;CONFIG DATA LOC 6
CFIGD7    EQU       46H            ;CONFIG DATA LOC 7
CFIGD8    EQU       47H            ;CONFIG DATA LOC 8
;
;
          ORG       0000H          ;
          LJMP      START          ;
;
```

Assembly Program Listing 3 (Cont.)

```
         ORG      0100H           ;
START:   MOVIE,#00H               ;DISABLE INTERUPTS
         CALL     LDCFGAD         ;LOAD CONFIGURATION ADDRESSES
         MOV      P1,#0FFH        ;SET PORTS 1 AND 3 ALL HIGH
         MOV      P3,#0FFH        ;
         CLR      P3.3            ;STROBE D/~P PIN LOW
         CLR      P3.2            ;STROBE ~RESET LOW
         SETB     P3.2            ;RELEASE ~RESET PIN
         SETB     P3.3            ;RELEASE D/~P PIN
         CALL     DLAY            ;GO TO DELAY ROUTINE
;
ENDTST:  MOV      A,CFGAD1        ;TEST FOR END OF DOWNLOAD
         CJNE     A,#46,NXTBYT    ;CHECK FOR 600HEX DATA
         SJMP     CFGDNE          ;SENT IF YES CONFIG DONE
NXTBYT:  CALL     CONFIG          ;ELSE CALL CONFIG ROUTINE
         MOV      R0,#08H         ;SET LOOP COUNT OF EIGHT
NXTBIT:  MOV      B,#00H          ;CLR B REGISTER GET
         MOV      A,CFIGD1        ;CONFIGURTION BYTE FOR
         RRC      A               ;LCA 1 AND ROTATE INTO
         MOV      CFIGD1,A        ;CARRY REGISTER. CHECK FOR
         JNC      NOBIT1          ;BIT SET, IF NO LEAVE BIT
         SETB     B.0             ;LOCATION CLEAR ELSE SET
NOBIT1:  MOV      A,CFIGD2        ;LOCATION. GET CONFIGURATION
         RRC      A               ;BYTE FOR LCA 2, ROTATE
         MOV      CFIGD2,A        ;INTO CARRY REGISTER IF
         JNC      NOBIT2          ;NOT SET LEAVE BIT LOCATION
         SETB     B.1             ;CLEAR ELSE SET BIT
NOBIT2:  MOV      A,CFIGD3        ;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 3
         MOV      CFIGD3,A        ;AND RESTORE AFTER
         JNC      NOBIT3          ;SHIFTING
         SETB     B.2             ;
NOBIT3:  MOV      A,CFIGD4        ;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 2
         MOV      CFIGD4,A        ;AND RESTORE AFTER
         JNC      NOBIT4          ;SHIFTING
         SETB     B.3             ;SET BIT 3
NOBIT4:  MOVA,CFIGD5;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 5
         MOV      CFIGD5,A        ;AND RESTORE AFTER
         JNC      NOBIT5          ;SHIFTING
         SETB     B.4             ;SET BIT 4
NOBIT5:  MOV      A,CFIGD6        ;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 6
         MOV      CFIGD6,A        ;AND RESTORE AFTER
         JNC      NOBIT6          ;SHIFTING
         SETB     B.5             ;SET BIT 5
NOBIT6:  MOV      A,CFIGD7        ;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 7
         MOV      CFIGD7,A        ;AND RESTORE AFTER
         JNC      NOBIT7          ;SHIFTING
         SETB     B.6             ;SET BIT 6
NOBIT7:  MOVA,CFIGD8;GET CONFIGURATION BYTE
         RRC      A               ;FOR LCA 8
```

Assembly Program Listing 3 (Cont.)

```
         MOV     CFIGD8,A         ;AND RESTORE AFTER
         JNC     DONE1            ;SHIFTING
         SETB    B.7              ;SET BIT 7
DONE1:   MOV     P1,B             ;SEND CONFIGURATION
         CLR     P3.5             ;DATA TO PORT P1
         SETB    P3.5             ;STROBE CCLK TO LOAD
         DEC     R0               ;EIGHT DATA BITS IN
         CJNE    R0,#00H,NXTBIT ;PARRALLEL. CHECK FOR
         SJMP    ENDTST           ;ONE BYTE SENT
CFGDNE:  JNB     P3.3,START       ;TEST THE DONE/PROGRAM
         MOV     P1,#0FFH         ;PIN IF HIGH SET PORT 1
         SJMP    PGMST            ;WITH ALL HIGH AND
;                                 ;GOTO MAIN PROGRAM
DLAY:    MOV     R0,07FH          ;DELAY LOOP TO ALLOW
IDEL:    DEC     R0               ;THE CONFIGURATION
         CJNE    R0,#00H,IDEL     ;CIRCUITRY TO RESET
         RET                      ;ADEQUATELY.
;
CONFIG:  MOV     R0,#CFIGD1       ;GET LCA 1 CONFIG BYTE
         MOV     R1,#CFGAD1       ;AND LCA 1 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD2       ;GET LCA 2 CONFIG BYTE
         MOV     R1,#CFGAD2       ;AND LCA 2 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD3       ;GET LCA 3 CONFIG BYTE
         MOV     R1,#CFGAD3       ;AND LCA 3 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD4       ;GET LCA 4 CONFIG BYTE
         MOV     R1,#CFGAD4       ;AND LCA 4 HIGH ADDRES
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD5       ;GET LCA 5 CONFIG BYTE
         MOV     R1,#CFGAD5       ;AND LCA 5 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD6       ;GET LCA 6 CONFIG BYTE
         MOV     R1,#CFGAD6       ;AND LCA 6 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD7       ;GET LCA 7 CONFIG BYTE
         MOV     R1,#CFGAD7       ;AND LCA 7 HIGH ADDRESS
         CALL    GETDAT           ;BYTE
         MOV     R0,#CFIGD8       ;GET LCA 8 CONFIG BYTE
         MOV     R1,#CFGAD8       ;AND LCA 8 HICH ADDRESS
         CALL    GETDAT;BYTE
         RET                      ;
;
GETDAT:  MOV     DPH,@R1          ;GET CURRENT ADDRESS
         INC     R1               ;POINTER OF THE CONFIG
         MOV     DPL,@R1          ;DATA INTO DPTR REGISTER
         DEC     R1               ;RESTORE UPPER ADDRESS
         CLR     A                ;POINTER
         MOVC    A,@A+DPTR        ;GET CONFIGURATION BYTE
         MOV     @R0,A            ;AND STORE IN THE CFIGDX
         INC     DPTR             ;RAM LOCATION
         MOV     @R1,DPH          ;INCREMENT POINTER TO
         INC     R1               ;ACCESS THE NEXT CONFIG
```

Assembly Program Listing 3 (Cont.)

```
         MOV      @R1,DPL        ;BYTE AND RESTORE THE
         RET                     ;ADDRESS POINTER
;
LDCFGAD: MOV      CFGAD1,#40H    ;LOAD ALL THE CONFIGURATION
         MOV      LOWAD1,#00H    ;ADDRESSES AT 4000H FOR
         MOV      CFGAD2,#46H    ;LCA 1 4600H FOR LCA 2
         MOV      LOWAD2,#00H    ;
         MOV      CFGAD3,#4CH    ;4C00H FOR LCA 3
         MOV      LOWAD3,#00H    ;
         MOV      CFGAD4,#52H    ;5200H FOR LCA 4
         MOV      LOWAD4,#00H    ;
         MOV      CFGAD5,#58H    ;5800H FOR LCA 5
         MOV      LOWAD5,#00H    ;
         MOV      CFGAD6,#5EH    ;5E00H FOR LCA 6
         MOV      LOWAD6,#00H    ;
         MOV      CFGAD7,#64H    ;6400H FOR LCA 7
         MOV      LOWAD7,#00H    ;
         MOV      CFGAD8,#6AH    ;6A00H FOR LCA 8
         MOV      LOWAD8,#00H    ;
         RET                     ;
;
;        MAIN PROGRAM CAN START HERE.
;
PGMST:
;
END
```

Introduction 1

Applications 2

Product Information 3

Table of Contents

AmPGA081

Configuration PROM Programmer

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Allows uploading, downloading, and saving of PROM files**
- **Can program and verify patterns**
- **Programs AMD and Xilinx™ Serial Configuration PROMs**
- **Programs 36K and 65K Serial Configuration PROMs**
- **Connects to serial port of IBM® PC-XT™, PC-AT™ or compatible**
- **Operates from PC via menu-driven software provided with the programming unit**
- **Accepts data files in any of the three XACT™-supported formats, MCS86, Tekhex and ExorMax**

GENERAL DESCRIPTION

The AmPGA081 Configuration PROM Programmer is designed to support the Am1736 and Am1765 Serial Configuration PROMs, (SCPs) 8-pin SKINNYDIP® PROMs used to configure programmable gate arrays. In addition, the AmPGA081 can also program the XC1736 device from Xilinx Corporation.

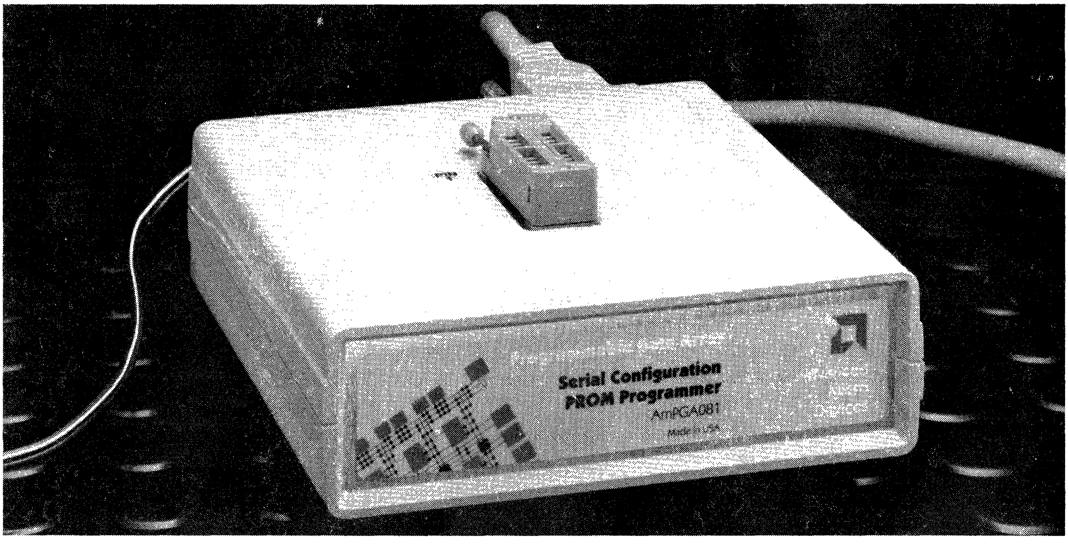
The programming unit, controlled using menu–driven, PC-based software, is connected to the serial port of an IBM PC-XT, PC-AT, or compatible. The AmPGA081 software and programmer are self-contained, and can be installed on a system other than that used for design development, such as one in a manufacturing area.

Designers compile their Logic Cell™ Array designs into a standard PROM format using the XACT Design System. The software provided with the AMPGA081 is then used to download the PROM file into the programming unit to program a SCP. PROM files can also be uploaded to the PC. SCPs already programmed can be verified, by comparing the device to a PROM file in RAM. The contents of the PROM can also be saved in RAM.

ORDERING INFORMATION

Further information is available from your local AMD sales office, authorized representative, or franchised distributor.

AmPGA081 **Configuration PROM Programmer**
Includes programming unit, power transformer, software and serial cable.



Xilinx, Logic Cell, and XACT are trademarks of the Xilinx Corporation.
PC–XT, AT are trademarks and IBM is a registered trademark of International Business Machines Corporation.
SKINNYDIP is a registered trademark of Advanced Micro Devices Inc.

Publication # 12527 Rev. A Amendment /0
Issue Date: August 1989

Preliminary

Am1736/Am1765

Serial Configuration PROM

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **One-time programmable 36,288 or 65, 536 x 1-bit serial memories designed to store configuration patterns for Logic Cell™ Arrays.**
- **Simple interface to the AMD LCA™ requires only one user I/O pin on the LCA.**
- **A single Am1736 supports all members of the Am2000 family and members of the Am3000 family as large as Am3042. The Am1765 supports all Am2000 and Am3000 family members. Both Serial Configuration PROMs support multiple patterns for daisy-chained configurations.**
- **Low power CMOS EPROM process.**
- **Cascadable to provide more memory for additional configurations or high-density arrays.**
- **Storage for multiple configurations for a single Logic Cell Array.**
- **Space-efficient, 8-pin ceramic DIP package**
- **Programming supported by leading programmer manufacturers.**

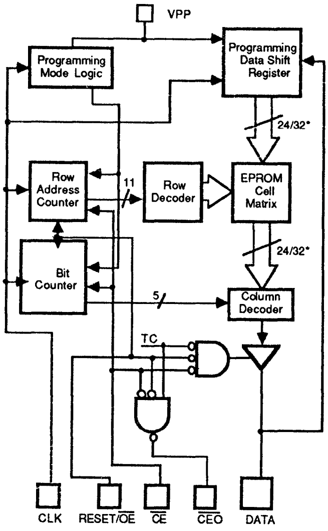
GENERAL DESCRIPTION

The Am1736 and Am1765 Serial Configuration PROMs (SCP) provide easy-to-use, cost-effective configuration memories for the AMD family of programmable gate arrays Packaged in an economical 8-pin DIP package, the devices use a simple serial access procedure to configure one or more Logic Cell Arrays (LCA). The 36,288 x 1-bit organization of the Am1736 supplies enough to configure any of the following devices: Am2064, Am2018, Am3020, and Am3042. The Am1765, with its 65,536 x 1-bits, in addition to the parts listed above, also supports the Am3040, Am3064 and Am3090 devices.

Multiple configurations for a single LCA device can be loaded from either SCP. Multiple SCPs can also be cascaded to provide larger memory for more configurations.

The Am1736/65 can be programmed on programming machines supplied by leading manufacturers, including Am081 from AMD. The LCA design file is first compiled into a standard HEX format with the PC-based Design System (AmPGA021). It can be transferred to the programmer through a serial port.

BLOCK DIAGRAM



*24-bit word for Am1736
32-bit word for Am1765

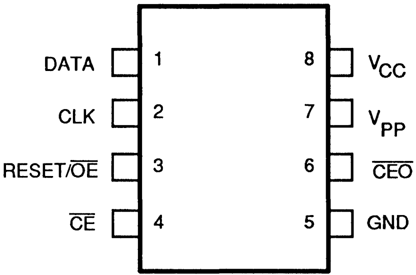
10867-001A

Logic Cell Array and LCA are trademarks of Xilinx Corporation.

Publication # 10867 Rev. B Amendment /0
Issue Date: August 1989

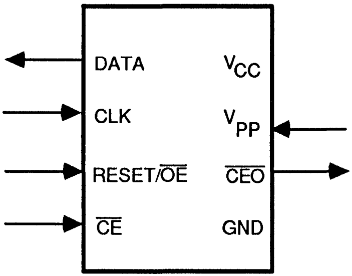
TABLE OF CONTENTS

CONNECTION DIAGRAM
8 Pin Ceramic DIP



10867-002A

LOGIC SYMBOL

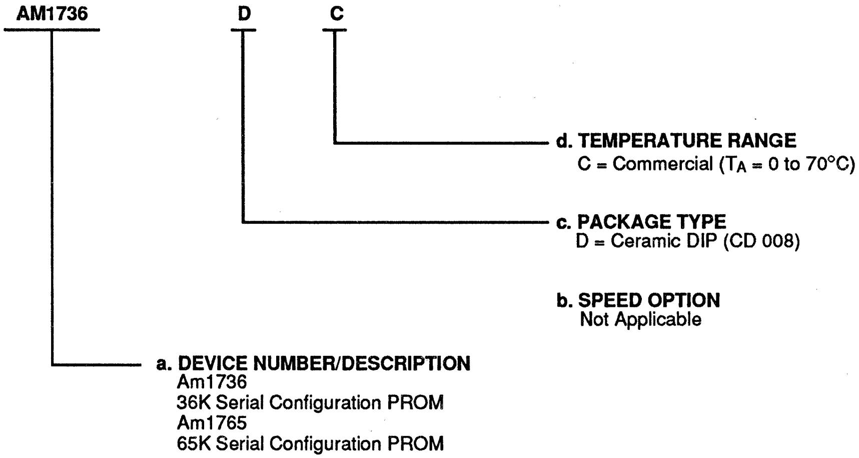


10867-015A

ORDERING INFORMATION
Standard Products

AMD standard products are available in several packages and operating ranges. The ordering number (Valid Combination) is formed by a combination of:

a. Device Number
b. Speed Option (if applicable)
c. Package Type
d. Temperature Range



AM1736 D C

d. TEMPERATURE RANGE
C = Commercial (T_A = 0 to 70°C)

c. PACKAGE TYPE
D = Ceramic DIP (CD 008)

b. SPEED OPTION
Not Applicable

a. DEVICE NUMBER/DESCRIPTION
Am1736
36K Serial Configuration PROM
Am1765
65K Serial Configuration PROM

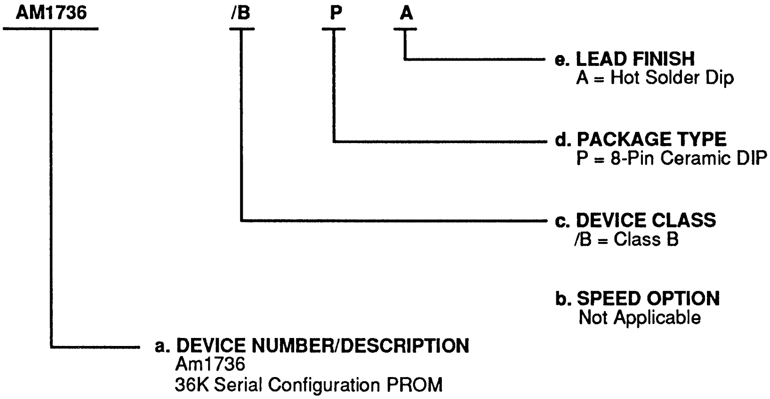
Valid Combination	
AM1736	DC
AM1765	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

ORDERING INFORMATION
Military Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of: **a. Device Number**
b. Speed Option (if applicable)
c. Device Class
d. Package Type
e. Lead Finish



AM1736 **/B** **P** **A**

e. LEAD FINISH
A = Hot Solder Dip

d. PACKAGE TYPE
P = 8-Pin Ceramic DIP

c. DEVICE CLASS
/B = Class B

b. SPEED OPTION
Not Applicable

a. DEVICE NUMBER/DESCRIPTION
Am1736
36K Serial Configuration PROM

Valid Combinations	
AM1736	/BPA

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consists of Subgroups 1,2,3,7,8,9,10,11.

PIN DESCRIPTIONS

DATA

Three-state DATA output for reading. Input/output pin for programming.

CLK

Clock Input

Used to increment the internal address and bit counters for reading and programming.

Reset/$\overline{\text{OE}}$

Output enable Input

A LOW level on both the $\overline{\text{CE}}$ and $\overline{\text{OE}}$ inputs enables the DATA output pin. A HIGH level on the $\overline{\text{OE}}$ pin resets both the address and bit counters.

$\overline{\text{CE}}$

Chip enable Input

A LOW level on both the $\overline{\text{CE}}$ and $\overline{\text{OE}}$ inputs enables the DATA output pin. A HIGH level on the $\overline{\text{CE}}$ pin disables both the address and bit counters and forces the device into a low power mode. Used for device selection.

GND

Ground pin.

$\overline{\text{CEO}}$

Chip enable output

This signal is asserted LOW on the clock cycle following the last bit read from memory. It will stay LOW as long as $\overline{\text{CE}}$ and $\overline{\text{OE}}$ are both LOW. It will follow $\overline{\text{CE}}$, but if $\overline{\text{OE}}$ goes HIGH, $\overline{\text{CEO}}$ will stay HIGH until the entire PROM is read again.

V_{PP}

Programming voltage supply. Used to enter the programming mode (+5 V) and to program memory (+12.5 V). Must be connected directly to V_{CC} for normal read operation. No overshoot above 13V permitted.

V_{CC}

+5 volt power supply.

LCA MASTER SERIAL MODE SUMMARY

The I/O and logic functions of the LCA, AMD's first programmable gate array, and their associated interconnections, are established by a configuration program. The program is loaded either automatically on power-up, or on command, depending on the state of the LCA mode pins. In Master Mode, the Logic Cell Array automatically loads the configuration program from an external memory device. The Serial Configuration PROM has been designed for compatibility with the Master Serial Mode.

On power-up or upon reconfiguration, an LCA will enter the Master Serial Mode whenever all three of the LCA's mode select pins are LOW (M0=0, M1=0, M2=0). Data is read from the Serial Configuration PROM sequentially on a single data line. Synchronization is provided by the rising edge of the CCLK signal, which is generated by the LCA during configuration.

Master Serial Mode provides a simple configuration interface. Only one serial data line and two control lines are needed to configure an LCA from the SCP. Data from the Serial Configuration PROM is read sequentially, accessed by the address and bit counters internal to the PROM that are incremented on every valid rising edge of the CLK input.

Programming the LCA with Counters Reset Upon Completion

Figure 3 shows the connections between an LCA and the SCP. The DATA line from the SCP is connected to the DIN input of the LCA. CCLK is connected to the CLK input of the SCP. At power-up or upon reconfiguration, the D/$\overline{P}$ signal is asserted low, enabling the SCP and its DATA output. During the configuration process, CCLK will clock data out of the SCP on every rising edge. At the completion of configuration, the D/$\overline{P}$ signal will switch high and reset the internal address counters of the SCP.

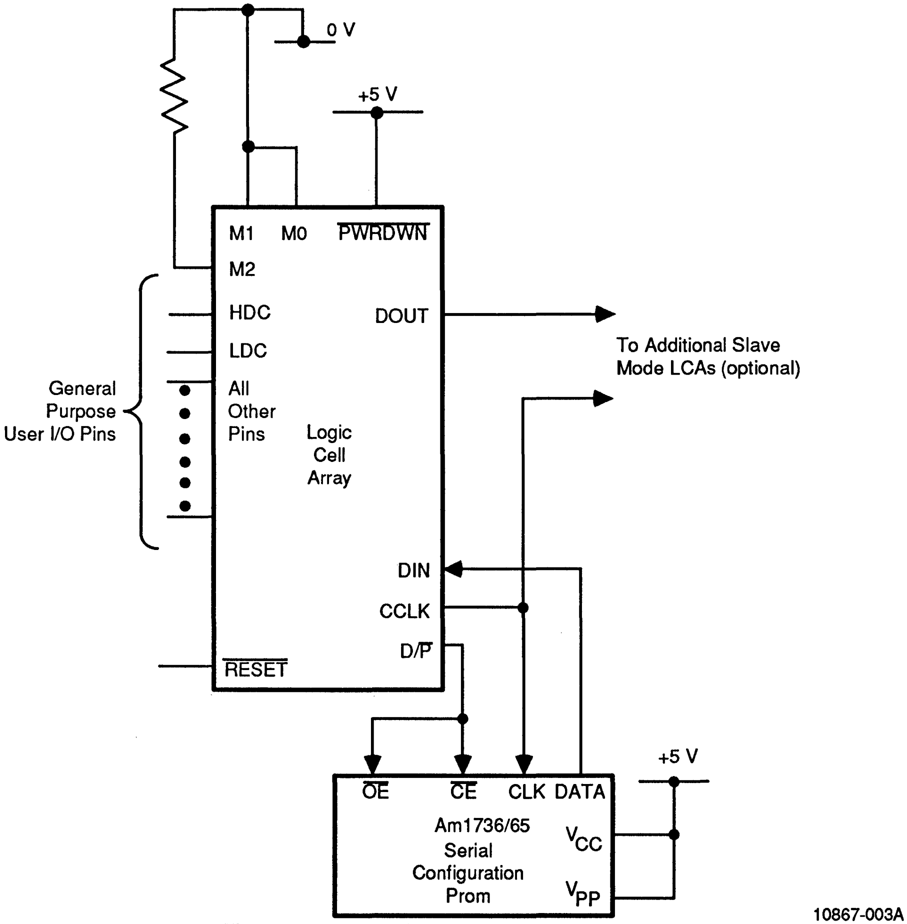


Figure 3. Master Serial Mode Control

If the user-programmable dual-function DIN pin of the LCA is used only for the configuration process, it should be programmed on the LCA so that no nodes are floating or in contention. For example, DIN can be programmed as an input, with an external pullup resistor attached.

If DIN is to be used for another function after configuration, contention between outputs must be avoided. By using the $\overline{\text{LDC}}$ pin to control the SCP's $\overline{\text{CE}}$ input, the SCP's DATA output is disabled one clock cycle before D/$\overline{\text{P}}$ switches high. If the LCA is to be reprogrammed after power-up, note that the LCA requires several microseconds to respond. In this case, by using $\overline{\text{LDC}}$ instead of D/$\overline{\text{P}}$ to control the SCP, the SCP will be enabled when the LCA has acknowledged the reprogram request, not several microseconds before. If $\overline{\text{LDC}}$ is used as a control signal to the SCP, it should be programmed on the LCA as an output high during normal operation.

Aborting the Configuration Process with RESET

If you wish to abort the configuration process by asserting $\overline{\text{RESET}}$ to the LCA, you must also reset the internal counters on the SCP. By connecting the $\overline{\text{RESET}}$ signal to the SCP's $\overline{\text{OE}}$ pin through an inverter as shown in Figure 4, the SCP will reset its internal counters. $\overline{\text{RESET}}$ must be used since neither $\overline{\text{LDC}}$ nor D/$\overline{\text{P}}$ switches high during an aborted configuration.

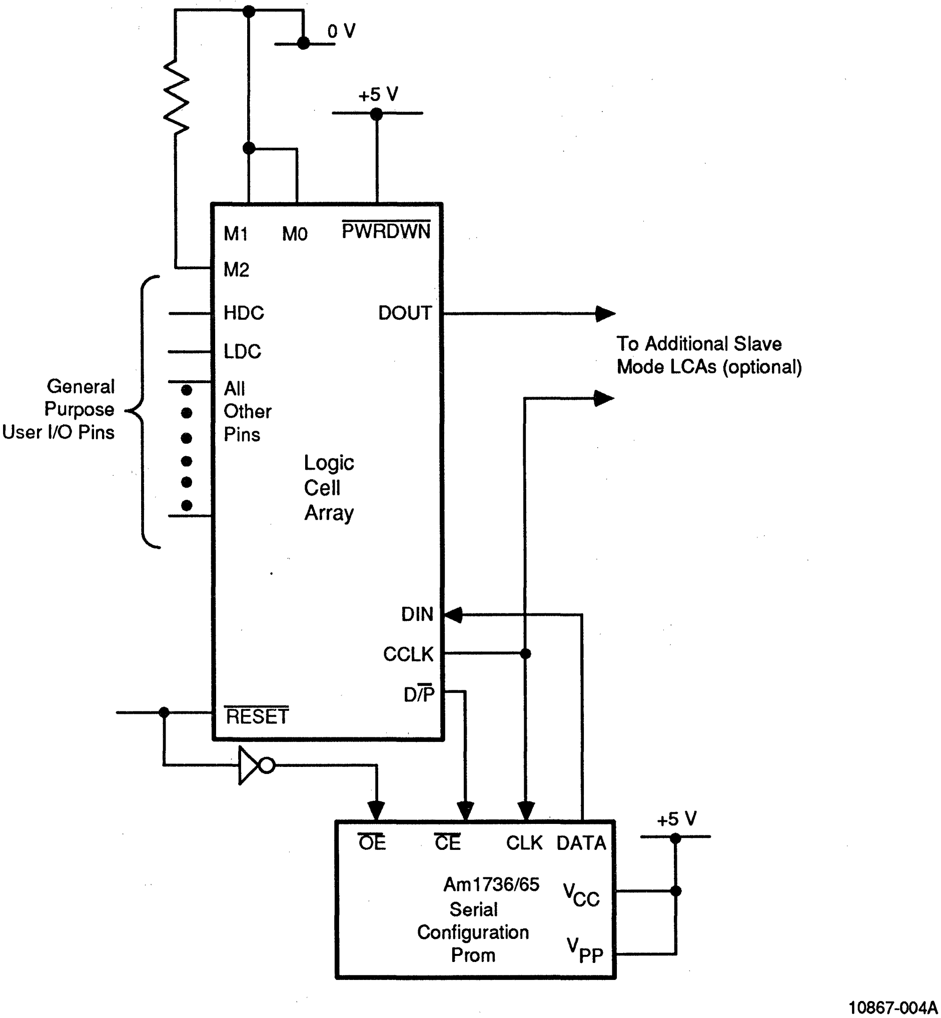
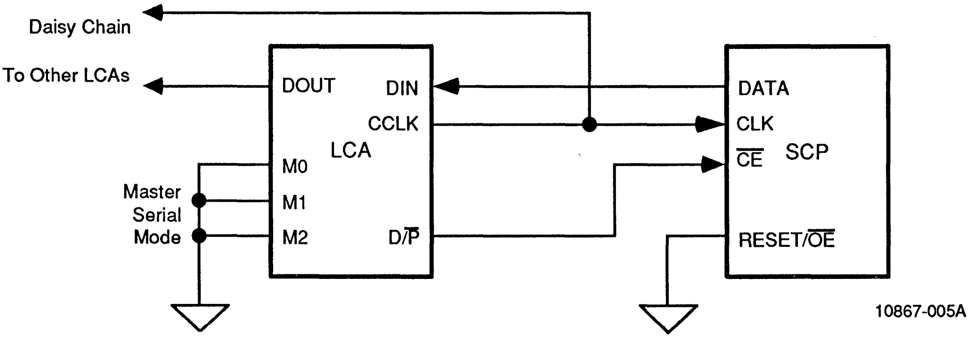


Figure 4. Aborting The Configuration Process With $\overline{\text{RESET}}$

Programming the LCA with Counters Unchanged upon Completion

When multiple LCA configurations for a single LCA are stored in a Serial Configuration PROM, the $\overline{OE}$ pin should be tied low as shown in Figure 5. Upon power-up, the internal address counters will be reset and configuration will begin with the first program stored in memory. Since $\overline{OE}$ is held low, the address counters are left unchanged after configuration is complete. Therefore, to reprogram the LCA with another program, the D/$\overline{P}$ line is pulled low, and configuration begins from the last value contained in the address counters.

You must not abort the configuration process when storing multiple LCA configurations in the SCP. When a configuration process is aborted and the internal counters of the SCP reset, the counters will always reset to address 0. It is not possible to reset the counters of the SCP to the beginning of the second or third configuration bitstream. Thus, when reconfiguring an LCA with other than the first pattern in the SCP, the configuration process must be allowed to finish.



Note: If M2 is tied directly to ground, it should be programmed as an input during operation.

Figure 5. Address Counters are not Reset after Configuration.

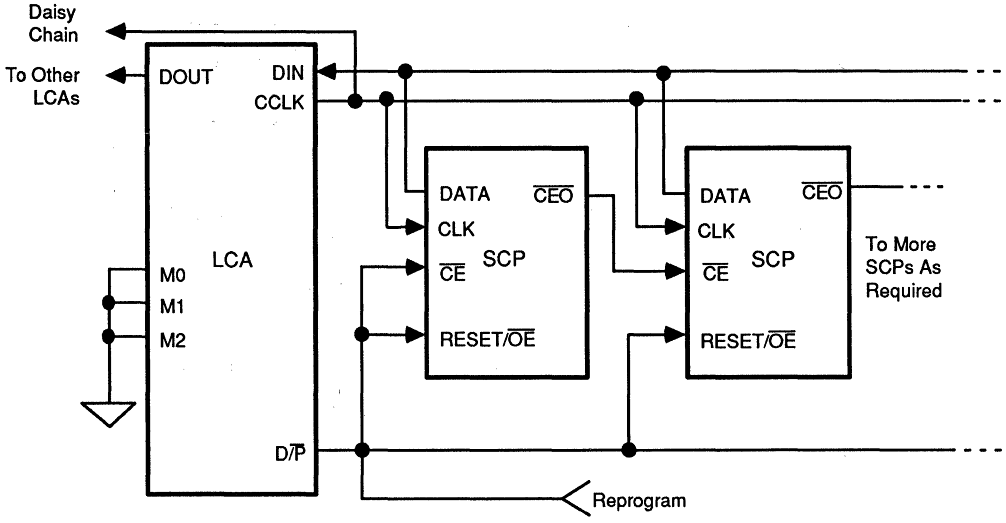
Cascading Serial Configuration PROMs

Am1736 supports devices up to and including an Am3042. The Am1765 supports all devices in the Am2000 and Am3000 family. However, for multiple configurations, it might be necessary to cascade SCPs together to provide for more configuration memory.

After the last bit from the first SCP is read, the SCP asserts its $\overline{\text{CEO}}$ output low and disables its own DATA line. The next SCP recognizes the low level on its $\overline{\text{CE}}$ input, and enables its own DATA output. See Figure 6.

After configuration is complete, the address counters of all the cascaded SCPs will be reset when the D/$\overline{\text{P}}$ pin goes high forcing the $\overline{\text{RESET}}/\overline{\text{OE}}$ on each SCP to go high.

If the address counters are not to be reset upon completion, then the $\overline{\text{OE}}$ inputs can be tied to ground, as previously shown in Figure 6. To reprogram LCAs configured by cascaded SCPs with another program, switch the D/$\overline{\text{P}}$ line low. Configuration begins where the address counters have stopped as D/$\overline{\text{P}}$ asserts $\overline{\text{CE}}$ on the SCPs.



10867-006A

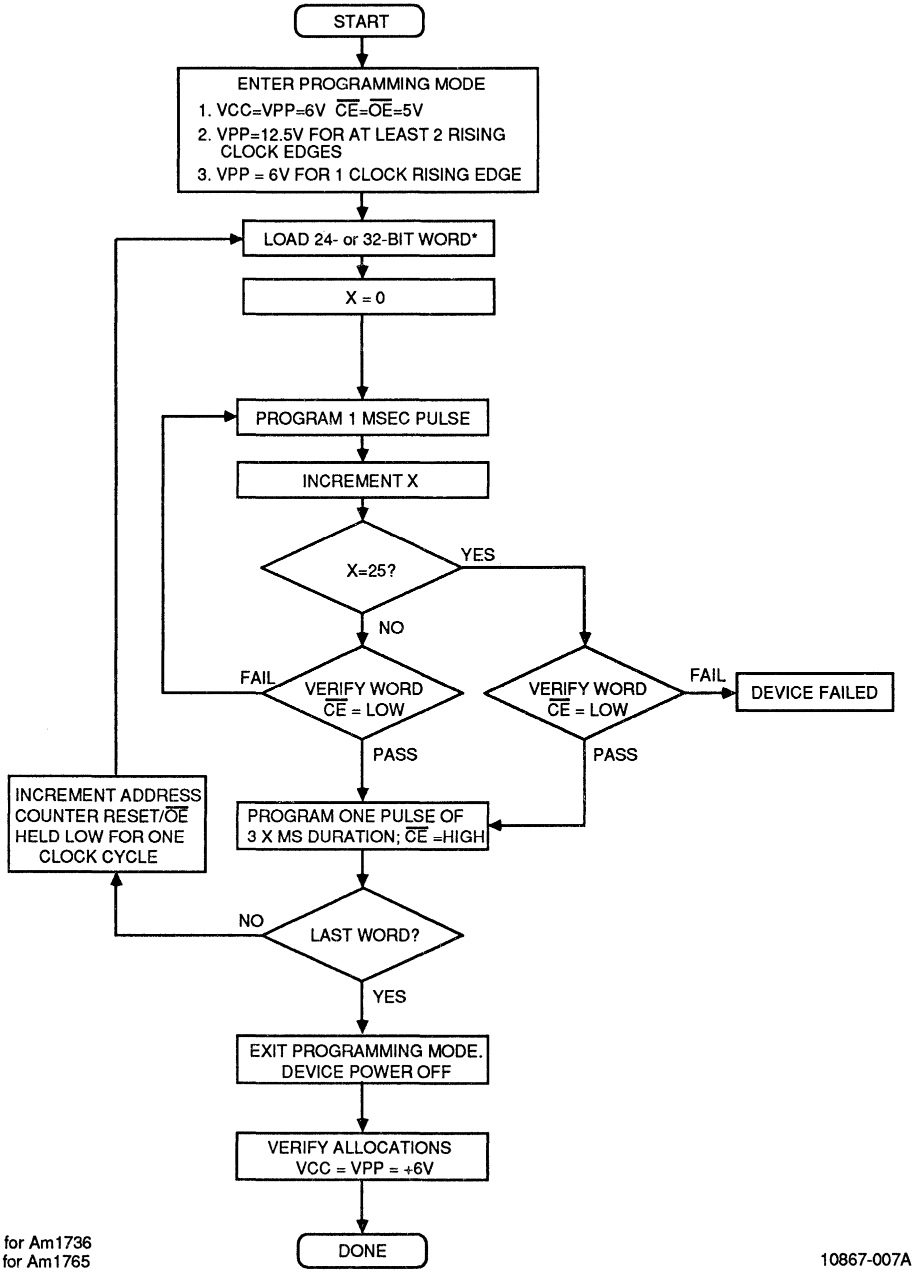
Figure 6. Cascading SCPs

Standby Mode

The Am1736/65 enters a low-power standby mode whenever $\overline{CE}$ is asserted high. In this mode, the SCP will consume less than 0.5mA of current. The output remains in a high-impedance state regardless of the state of the $\overline{OE}$ input.

Programming Mode

Figure 7 shows the programming algorithm for the Am1736/65. Note that the programming mode is entered by holding V_{PP} high for at least two clock edges and is exited by removing power from the device.


START
ENTER PROGRAMMING MODE
1. VCC=VPP=6V $\overline{CE}$=$\overline{OE}$=5V
2. VPP=12.5V FOR AT LEAST 2 RISING CLOCK EDGES
3. VPP = 6V FOR 1 CLOCK RISING EDGE
LOAD 24- or 32-BIT WORD*
X = 0
PROGRAM 1 MSEC PULSE
INCREMENT X
X=25?
YES
NO
FAIL
VERIFY WORD $\overline{CE}$ = LOW
VERIFY WORD $\overline{CE}$ = LOW
FAIL
DEVICE FAILED
PASS
PASS
INCREMENT ADDRESS COUNTER RESET/$\overline{OE}$ HELD LOW FOR ONE CLOCK CYCLE
PROGRAM ONE PULSE OF 3 X MS DURATION; $\overline{CE}$ =HIGH
NO
LAST WORD?
YES
EXIT PROGRAMMING MODE. DEVICE POWER OFF
VERIFY ALLOCATIONS VCC = VPP = +6V
DONE

*24-bit word for Am1736
32-bit word for Am1765

10867-007A

Figure 7. Programming Mode

ABSOLUTE MAXIMUM RATINGS

Supply voltage relative to GND (V_{CC})	-0.5 V to +7.0 V
Supply voltage relative to GND (V_{PP})	-0.5 V to +13.5 V
Input voltage with respect to GND (V_{IN})	-0.5 V to V_{CC} + 0.5 V
Voltage applied to three-state output (V_{TS})	-0.5 V to V_{CC} + 0.5 V
Storage temperature (ambient) (T_{STG})	-65 to +125°C
Maximum soldering temperature (10 seconds at 1/16 inch) (T_{SOL})	+260°C

OPERATING RANGES

Commercial (C) devices	
Supply Voltage	4.75 V to 5.25 V
Ambient Temperature (T_A)	0 to +70°C
Industrial (I) devices	
Supply Voltage	4.5 V to 5.5 V
Ambient Temperature (T_A)	-40 to +85°C
Military (M) devices	
Supply Voltage	4.5 V to 5.5 V
Ambient Temperature (T_A)	-55 to +125°C

Note: Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those listed under Recommended Operating Conditions is not implied. Exposure to Absolute Maximum Ratings for extended periods of time may affect device reliability.

DC CHARACTERISTICS

Symbol	Description	Test Conditions		Min	Max	Unit
V_{IH}	High-level input voltage			2.0	V_{CC}	V
V_{IL}	Low-level input voltage			0	0.8	V
V_{OH}	High-level output voltage	(I_{OH} = -4mA)	Commercial	3.86		V
V_{OL}	Low-level output voltage	(I_{OL} = 4mA)			0.32	V
V_{OH}	High-level output voltage	(I_{OH} = -4mA)	Industrial	3.76		V
V_{OL}	Low-level output voltage	(I_{OL} = 4mA)			0.37	V
V_{OH}	High-level output voltage	(I_{OH} = -4mA)	Military	3.7		V
V_{OL}	Low-level output voltage	(I_{OL} = 4mA)			0.4	V
ICC_A	Supply current, active mode	(I_{CC} + I_{PP})	$\overline{CE}$ = $\overline{OE}$ =GND			
	Am1736				10	mA
	Am1765				20	mA
ICC_S	Supply current, standby mode	(I_{CC} + I_{PP})	$\overline{CE}$ =V_{CC}			
	Am1736				0.5	mA
	Am1765				1	mA
I_L	Input or output leakage current				±10	µA

AC CHARACTERISTICS

Symbol		Description	Limits		Unit
			Min	Max	
1	T_{OE}	$\overline{OE}$ to data enable delay		100	ns
2	T_{CE}	$\overline{CE}$ to data enable delay		250	ns
3	T_{CAC}	CLK to data delay		400	ns
4	T_{OH}	Data hold from $\overline{CE}$, $\overline{OE}$, or CLK	0		ns
5	T_{DF}	$\overline{CE}$ or $\overline{OE}$ to data disable delay		50	ns
6	T_{LC}	CLK low time	200		ns
7	T_{HC}	CLK high time	200		ns
8	T_{SCE}	$\overline{CE}$ setup time to CLK (Guarantees counters will or will not change)	100		ns
9	T_{HCE}	$\overline{CE}$ hold time to CLK (Guarantees counters will or will not change)	0		ns
10	T_{HOE}	$\overline{OE}$ high time (Guarantees counters are reset)	100		ns
11	T_{CDF}	CLK to data disable delay		40	ns
12	T_{OCK}	CLK to $\overline{CEO}$ delay		100	ns
13	T_{OCE}	$\overline{CE}$ to $\overline{CEO}$ delay		100	ns
14	T_{OOE}	RESET/ $\overline{OE}$ to $\overline{CEO}$ delay		100	ns

Input pulse amplitude 0 V to 3.0 V

LCA MASTER SERIAL MODE CHARACTERISTICS

Symbol		Description	Limits		Unit
			Min	Max	
1	T_{CCK}	V_{CC} = 3 V to CCLK delay[1]	10	160	ms
2	T_{LCK}	CCLK low time	250		ns
3	T_{HCK}	CCLK high time	250		ns
4	T_{SD}	DIN setup to CCLK	100		ns
5	T_{HD}	DIN hold from CCLK	0		ns
6	T_{DONE}	LCA User programmed pins active to done delay		3	µs

[1] This time may be extended by holding the LCA's $\overline{RESET}$ input low.

DC PROGRAMMING CHARACTERISTICS

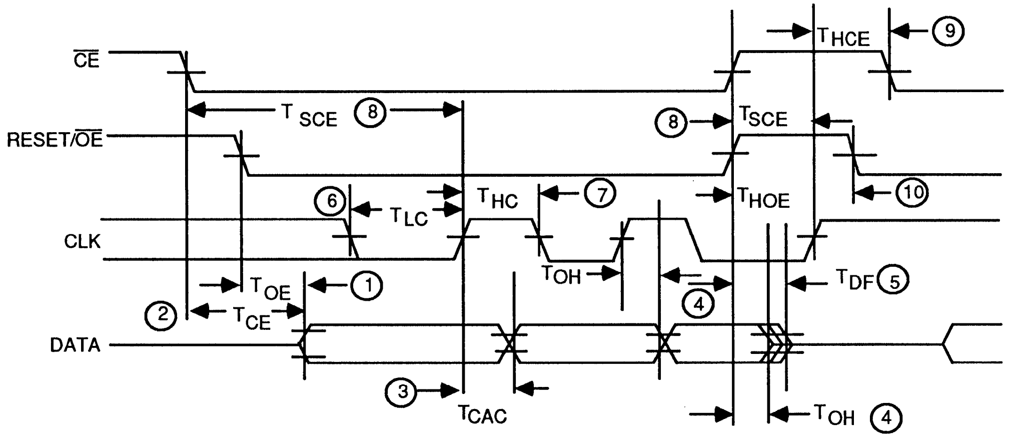
Symbol	Description	Test Conditions	Limits		Unit
			Min	Max	
I_{LI}	Input current (all inputs)	$V_{IN} = V_{IL}$ or V_{IH}		±10.0	µA
V_{IL}	Input LOW level (all inputs)		0.0	0.8	V
V_{IH}	Input HIGH level		2.0	V_{CC}	V
V_{OL}	Output LOW voltage during verify	I_{OL} = 4mA		0.4	V
V_{OH}	Output HIGH voltage during verify	I_{OH} = -4mA	3.7		V
I_{CC2}	V_{CC} supply current (program and verify)			10	mA
I_{PP2}	V_{PP} supply current (program and verify)			30	mA
V_{CC}	Supply voltage		5.75	6.25	V
V_{PP1}	Programming voltage		12	13	V
V_{PP2}	V_{PP} during programming mode		5.75	6.25	V

SWITCHING PROGRAMMING CHARACTERISTICS

Symbol		Description	Limits		Unit
			Min	Max	
1	T_{RPP}	10% to 90% rise time of V_{PP}	50	70	µs
2	T_{FPP}	90% to 10% fall time of V_{PP}	50	70	µs
3	T_{PGM}	V_{PP} Programming pulse width	0.95	1.05	ms
4	T_{SVC}	Setup of V_{PP} to CLK to enter programming mode	2		µs
5	T_{HVC}	Hold of V_{PP} to CLK to enter programming mode	2		µs
6	T_{SDP}	Data setup to CLK for programming	2		µs
7	T_{HDP}	Data hold to CLK for programming	0		ns
8	T_{SCC}*	$\overline{CE}$ setup to CLK for verifying in programming mode	2		µs
9	T_{HCC}*	$\overline{CE}$ hold from CLK for verifying in programming mode	2		µs
10	T_{SCV}	$\overline{CE}$ setup to V_{PP} for programming	2		µs
11	T_{HCV}	$\overline{CE}$ hold from V_{PP} for programming	2		µs
12	T_{SIC}	$\overline{OE}$ setup to CLK to increment address counter	2		µs
13	T_{HIC}	$\overline{OE}$ hold from CLK to increment address counter	0		ns

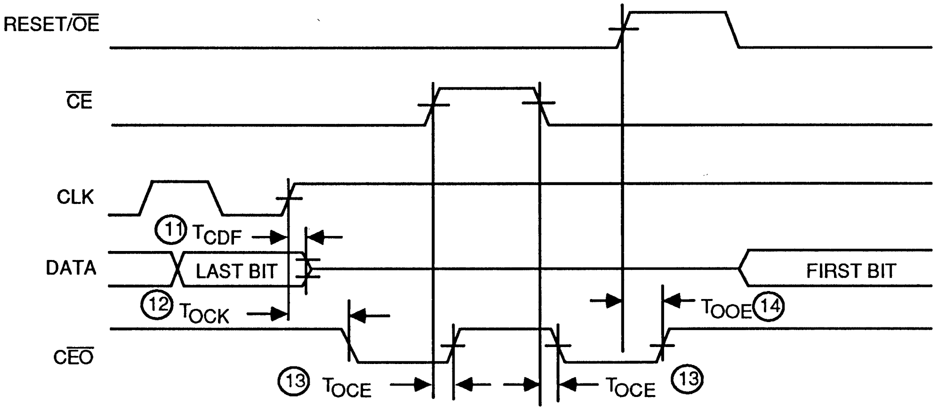
* During programming $\overline{CE}$ should only be changed while CLK is high and has been high for 200 ns.

SWITCHING WAVEFORMS



10867-009A

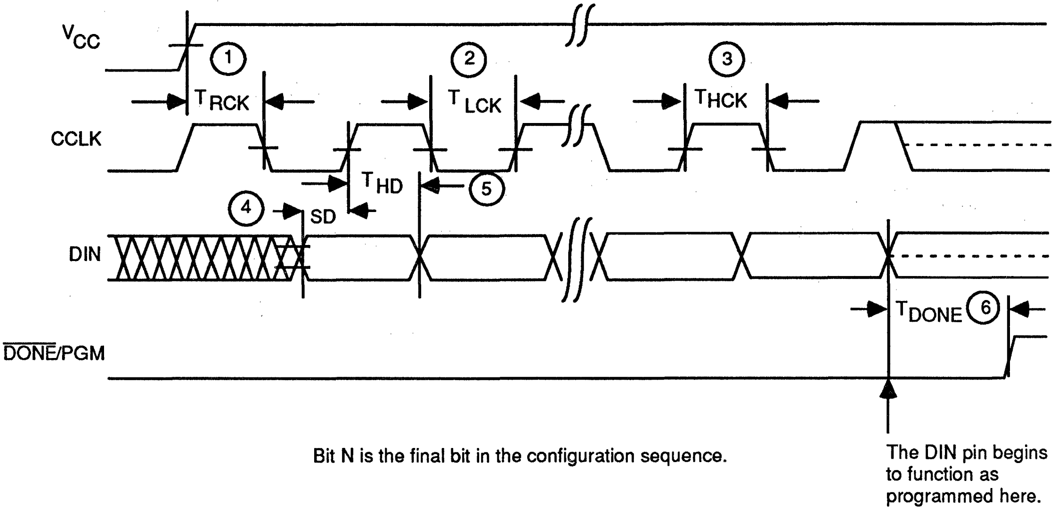
AC Characteristics



10867-010A

AC Characteristics

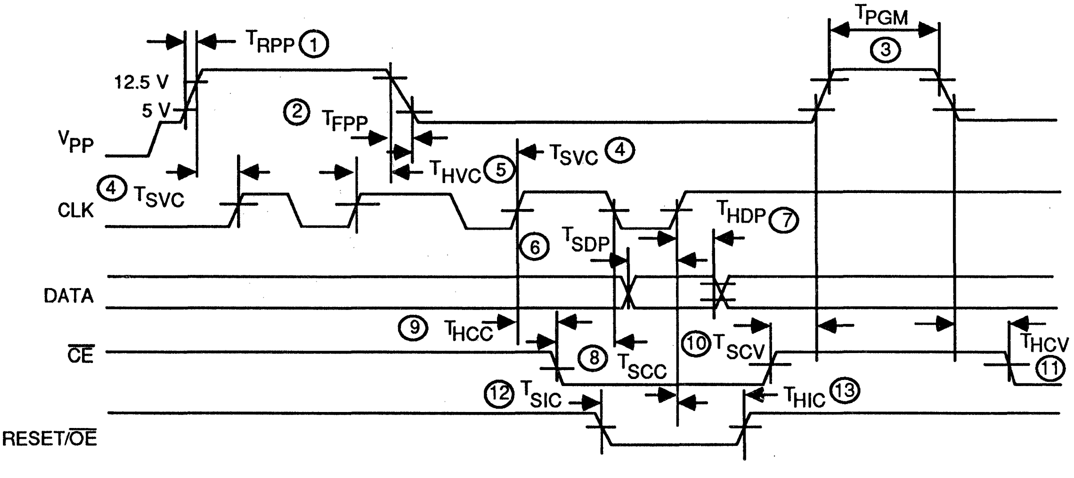
SWITCHING WAVEFORMS (Continued)



V_{CC}
CCLK
DIN
DONE/PGM
1
T_{RCK}
2
T_{LCK}
3
T_{HCK}
4
SD
T_{HD}
5
T_{DONE}
6
Bit N is the final bit in the configuration sequence.
The DIN pin begins to function as programmed here.

10867-011A

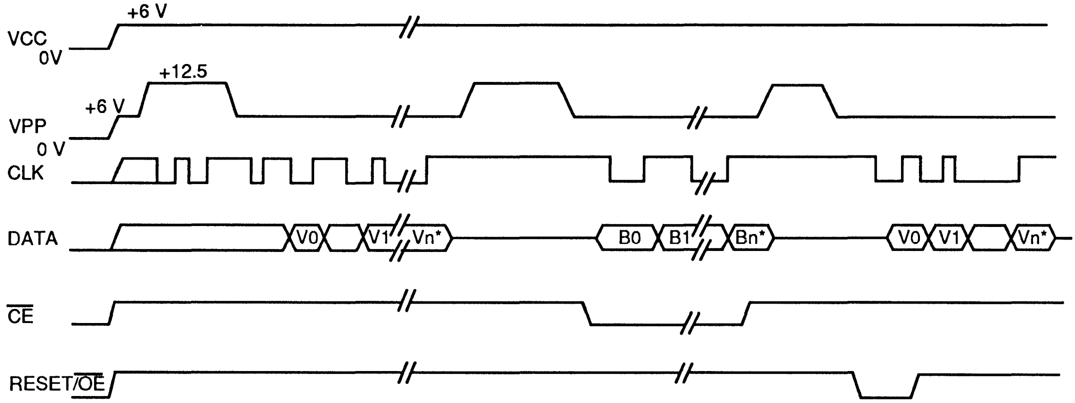
LCA Master Serial Mode Characteristics



10867-012A

Switching Programming Characteristics

SWITCHING WAVEFORMS (Continued)



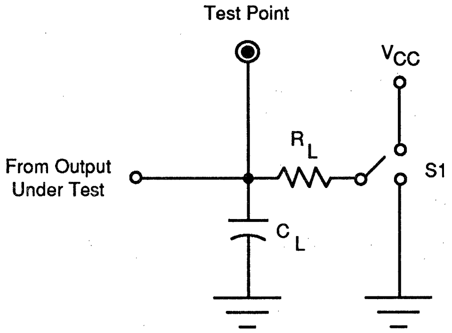
ENTER PROGRAMMING MODE	LOAD THE FIRST WORD	PROGRAMMING PULSE	VERIFY THE WORD	OVER PROGRAM PULSE	LOAD THE NEXT WORD; INCREMENT ADDRESS COUNTER
A minimum of two clock rising edges must occur while V_{PP} is at 12.5 volts to set the device ready for entering programming mode. Programming mode is entered on the first clock rising edge after V_{PP} goes to 5.0 V. To exit the programming mode, remove power from the device.	Each bit is loaded by applying a clock rising edge to the device until all bits have been loaded.	Program the word, that was previously loaded, into the first address of the array. Repeat the program-verify process (up to 25 times) until the whole word verify passes.	Note: It is not necessary to read all bits if verify fails.	This pulse is three times the sum of all pulses for this word.	

* n = 23 for Am1736, word size = 24 bits
  = 31 for Am1765, word size = 32 bits

10867-013A

Programming Waveforms

TEST CIRCUIT

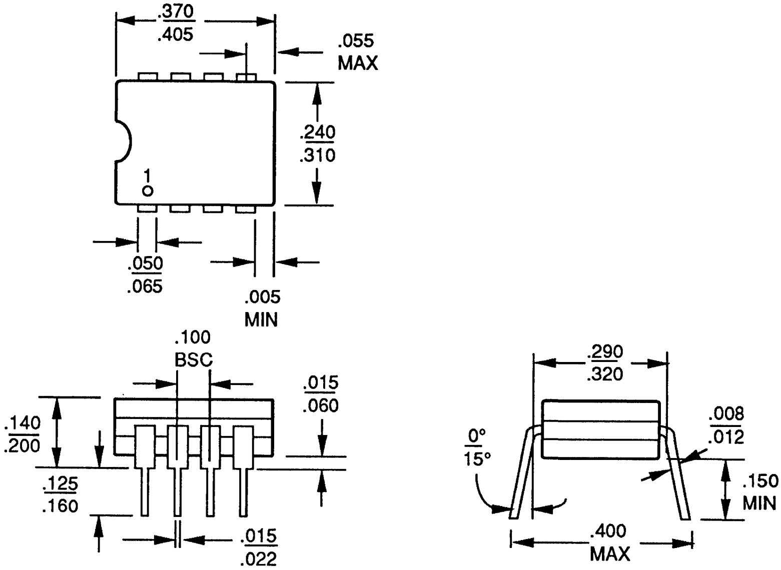


10867-008A

Test Circuit

Description		**Switch S1**	**Cl**	**Rl**	**Measured Output Value**
Active output propagation delays		Open	50 pF		0.5*V_{CC}
Input to output enable delays	Z-->H	GND	50 pF	1K	0.5*V_{CC}
	Z-->L	V_{CC}	50 pF	1K	0.5*V_{CC}
Input to output disable delays	H-->Z	GND	5 pF	1K	V_{OH} - 0.5 V
	L-->Z	V_{CC}	5 pF	1K	V_{OL} + 0.5 V

PHYSICAL DIMENSIONS
CD 008



12101A

3

For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

Am2064/Am2018

Programmable Gate Array

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Up to 1800 usable gate equivalence for higher board densities**
- **TTL or CMOS input threshold levels allow for more board flexibility**
- **Field programmable gate array means easy and quick design modifications**
- **Reconfigurable device for multiple in-system patterns**
- **Readback feature allows you to check configuration data**

GENERAL DESCRIPTION

The Logic Cell Array™ (LCA)™ is a high density, CMOS programmable gate array. It is composed of an interior array of logic blocks, surrounded by a ring of I/O interface blocks. Unlike conventional gate arrays, however, the definition of logic functions and interconnections in an LCA device does not require any custom factory fabrication. The configurable logic blocks (CLBs), I/O blocks (IOBs), and interconnection resources of the device are completely user-configurable. Each device is identical until it is configured. When configured, the LCA then begins operating according to the logic defined by the user. An LCA can either load its configuration data automatically from an external EPROM or have the data loaded under microprocessor control. The LCA is configured each time it is powered up, and offers the user the unique benefit of being able to reconfigure in-system at any time during operation.

The Logic Array is a CMOS, static RAM-based device. Static RAM technology is a mature technology that has been in production for years, with millions of operating device hours. Compared to other programming alternatives, static memory provides the best combination of high density, high performance, high reliability, and comprehensive testability. The LCA is the first device to apply the benefits of static RAM technology to overcome the disadvantages of conventional gate arrays. The static memory cells used for the configuration memory of the LCA have been designed specifically for high reliability and noise immunity. The memory cell of the LCA is much more stable than a typical high-speed conventional memory cell.

PGA development system software is available for all phases of the design cycle. From design entry through simulation, both logic and timing, automatic placement and routing, and in-circuit emulation, software that operates on an IBM®-PC-AT™ or compatible system speeds the design process. For users of Daisy™ or Mentor™ workstations, interfaces for PGA design are also available from AMD. A valid workstation interface is available from Valid Logic Systems.

Part Number	Logic Capacity (Usable Gates)	Configurable Logic Blocks	Configurable I/O Blocks	Configuration Program (Bits)
Am2064	1200	64	58	12,048
Am2018	1800	100	74	17,888

XILINX™· XACT™· XACTOR™· Logic Cell™ Array and LCA™ are trademarks of XILINX Inc. IBM ® is a registered trademark of International Business Machines Corporation. PC-AT™ is a trademark of International Business Machines Corporation. Daisy™ is a trademark of Daisy Corporation. Mentor ™ is a trademark of Mentor Corporation.

Portion of this Data Sheet reproduced with the permission of XILINX Inc.

Publication # Rev. Amendment
10352 B /0
Issue Date: May 1989

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

a. **Device Number**
b. **Speed Option** (if applicable)
c. **Package Type**
d. **Temperature Range**
e. **Number of Pins**

Am2018 - 50 J C 068

e. NUMBER OF PINS
028 (28 Pins)
044 (44 Pins)
048 (48 Pins)
068 (68 Pins)
084 (84 Pins)

d. TEMPERATURE RANGE
C = Commercial
I = Industrial

c. PACKAGE TYPE
P = Plastic DIP
J = Plastic Leaded Chip Carrier
G = Pin Grid Array

b. SPEED OPTION
-33 (33 MHz Toggle Rate)
-50 (50 MHz Toggle Rate)
-70 (70 MHz Toggle Rate)
-100 (100 MHz Toggle Rate)

a. PART NUMBER
2064 (1200 Gates)
2018 (1800 Gates)

PACKAGE AVAILABILTY

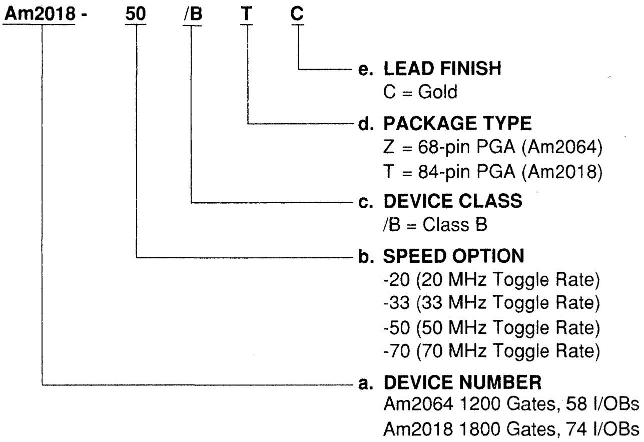
PART NUMBER	28-PIN PLCC	44-PIN PLCC	48-PIN PLASTIC DIP	68-PIN PLCC	68-PIN PGA	84-PIN PLCC	84-PIN PGA
Am2064	X		X	X	X		
Am2018		X		X	X	X	X

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

a. **Device Number**
b. **Speed Option** (if applicable)
c. **Device Class**
d. **Package Type**
e. **Lead Finish**



Am2018 - 50 /B T C

e. LEAD FINISH
C = Gold

d. PACKAGE TYPE
Z = 68-pin PGA (Am2064)
T = 84-pin PGA (Am2018)

c. DEVICE CLASS
/B = Class B

b. SPEED OPTION
-20 (20 MHz Toggle Rate)
-33 (33 MHz Toggle Rate)
-50 (50 MHz Toggle Rate)
-70 (70 MHz Toggle Rate)

a. DEVICE NUMBER
Am2064 1200 Gates, 58 I/OBs
Am2018 1800 Gates, 74 I/OBs

Valid Combinations	
Am2064-20 Am2064-33 Am2064-50	/BZC
Am2018-20 Am2018-33 Am2018-50 Am2018-70	/BTC

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, or to check on newly released valid combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A Tests consist of Subgroups: 1, 2, 3, 7, 8, 9, 10, 11.

SILICON MENU

AMD Part	Organ-ization	Equiva-lent Gate Count	Configu-rable Logic Block	User I/Os	Configu-ration Program Bits	Max Standby Current (CMOS Inputs)	Max Standby Current (TTL Inputs)	Packages	Max Toggle Rate Between CLBs
Am2064-20	8x8	1200	64	58	12048	10 mA	10 mA	68PGA	20 MHz
Am2018-20	10x10	1800	100	74	17888	15 mA	10 mA	84PGA	20 MHz
Am2064-33	8x8	1200	64	58	12048	10 mA	10 mA	68PGA	33 MHz
Am2018-33	10x10	1800	100	74	17888	15 mA	10 mA	84PGA	33 MHz
Am2064-50	8x8	1200	64	58	12048	10 mA	10 mA	68PGA	50 MHz
Am2018-50	10x10	1800	100	74	17888	15 mA	10 mA	84PGA	50 MHz
Am2018-70	10x10	1800	100	74	17888	15 mA	10 mA	84PGA	70 MHz

PIN DESCRIPTION

A. Permanently dedicated pins

1. V_{CC}

One or two (depending on package type) connections to the nominal +5 V supply voltage. All must be connected.

2. GND

One or two (depending on package type) connections to ground. All must be connected.

3. $\overline{\text{PWRDWN}}$

An active LOW power down input stops all internal activity to minimize V_{CC} power and puts all output buffers in a high impedance state. Configuration is retained, however, internal storage elements are reset. When the $\overline{\text{PWRDWN}}$ pin returns HIGH, the device returns to operation with the same sequence of buffer enable and DONE/$\overline{\text{PROGRAM}}$ as at the completion of configuration. If not used, $\overline{\text{PWRDWN}}$ must be tied to V_{CC}.

4. $\overline{\text{RESET}}$

This active LOW input has three functions. Prior to the start of configuration, a LOW input will delay the start of configuration. An internal circuit senses the application of power and begins a minimal time-out cycle. When the time-out and $\overline{\text{RESET}}$ are complete, the levels of the M lines are sampled and configuration begins. If $\overline{\text{RESET}}$ is asserted during a configuration, the LCA device is reinitialized and will restart the configuration at the termination of $\overline{\text{RESET}}$. If $\overline{\text{RESET}}$ is asserted after configuration is complete, it will provide an asynchronous reset of all IOB and CLB storage elements of the LCA device. $\overline{\text{RESET}}$ can also be used to recover from partial power failure. See section on Reprogram under "Special Features."

5. CCLK

During configuration, Configuration Clock is an output of an LCA device in master mode or peripheral mode. LCA devices in slave mode use it as a clock input. During a Readback operation, it is a clock input for the configuration data being shifted out.

6. DONE

The DONE output is configurable as an open drain with or without an internal pull-up resistor. At the completion of configuration, the circuitry of the LCA device becomes active in a synchronous order, and DONE can be programmed to occur one cycle before or after. Or:

PROG

Once configuration is completed, a HIGH-to-LOW transition of this pin will cause an initialization of the LCA device and start a reconfiguration.

7. M0

As Mode 0, this input, M1, and M2 are sampled before the start of configuration to establish the configuration mode to be used. Or:

RTRIG

As Read Trigger, after configuration is complete, an input transition to a HIGH will initiate a Readback of configuration and storage element data by CCLK. This operation can be limited to a single request, or can be inhibited altogether, by selecting the appropriate readback option when generating the bit stream.

8. M1

As Mode 1, this input, M0, and M2 are sampled before the start of configuration to establish the configuration mode, If readback is to be used, a 5-KΩ resistor should be used to define mode level inputs. Or:

RDATA

As an active LOW Read Data, this pin is the output of the readback data after configuration is complete.

B. User I/O Pins that can have special functions

1. M2

As Mode 2 this input has a passive pull-up during configuration. Together with M0 and M1, it is sampled before the start of configuration to establish the configuration mode. After configuration, this pin becomes a user-programmable I/O pin.

2. HDC

High During Configuration is held at a HIGH level by the LCA device until after configuration. It is available as a control indication that configuration is not completed. After configuration, this pin is a user I/O pin.

3. $\overline{\text{LDC}}$

Low During Configuration is held at a LOW level by the LCA device until after configuration. It is intended to be available as a control indication that configuration is not completed. It is particularly useful in master mode as a LOW enable for an EPROM. After configuration, this pin is a user I/O pin. If used as a LOW EPROM enable, it must be programmed as a HIGH after configuration.

4. XTL1

This user I/O pin can be configured to operate as the output of an amplifier driving an external crystal and bias circuitry.

5. XTL2

This user I/O pin can be configured to operate as the input of an amplifier usable with an external crystal and bias circuitry. This I/O block is left unconfigured. The oscillator configuration is activated by routing a net from the oscillator buffer symbol output and by the MakeBits program.

6. $\overline{\text{CS0}}$, $\overline{\text{CS1}}$, CS2, $\overline{\text{WRT}}$

These four inputs represent a set of signals, three active LOW and one active HIGH, which are used in peripheral mode to control configuration data entry. The assertion of all four generates a LOW CCLK. In master mode, these pins become part of the parallel configuration byte (D4, D3, D2, and D1). After configuration is complete, they are user-programmable I/O pins.

7. $\overline{\text{RCLK}}$

During master parallel mode configuration, this pin represents a read of an external memory device. (Normally not used.)

8. D0-D7

This set of eight pins represents the parallel configuration byte for the parallel master and peripheral modes. After configuration is complete, they are user-programmable I/O pins.

9. A0-A15

This set of 16 pins presents an address output for a configuration EPROM during master parallel mode. After configuration is complete, they are user-programmable I/O pins.

10. DIN

This user I/O pin is used as serial Data In during slave or master serial configuration. This pin is Data 0 input master or peripheral configuration mode.

11. DOUT

This user I/O pin is used during configuration to output serial configuration data for the Data In of daisy-chained slaves.

C. Unrestricted user I/O pins

1. I/O

A pin that, after configuration, can be programmed by the user to be an input and/or output pin. Some of these pins present a high impedance pull-up or perform other functions before configuration is complete.

FUNCTIONAL DESCRIPTION

The general structure of a Logic Cell Array is shown in Figure 1. The elements of the array include three categories of user-programmable elements: I/O Blocks, Configurable Logic Blocks and Programmable Interconnections. The I/O Blocks provide an interface between the logic array and the device package pins. The Configurable Logic Blocks perform user-specified logic functions, and the interconnect resources are programmed to form networks that carry logic signals among blocks.

Configuration of the Logic Cell Array is established through a distributed array of memory cells. The PGA Development System generates the program used to configure the Logic Cell Array. The Cell Array includes logic to implement automatic configuration.

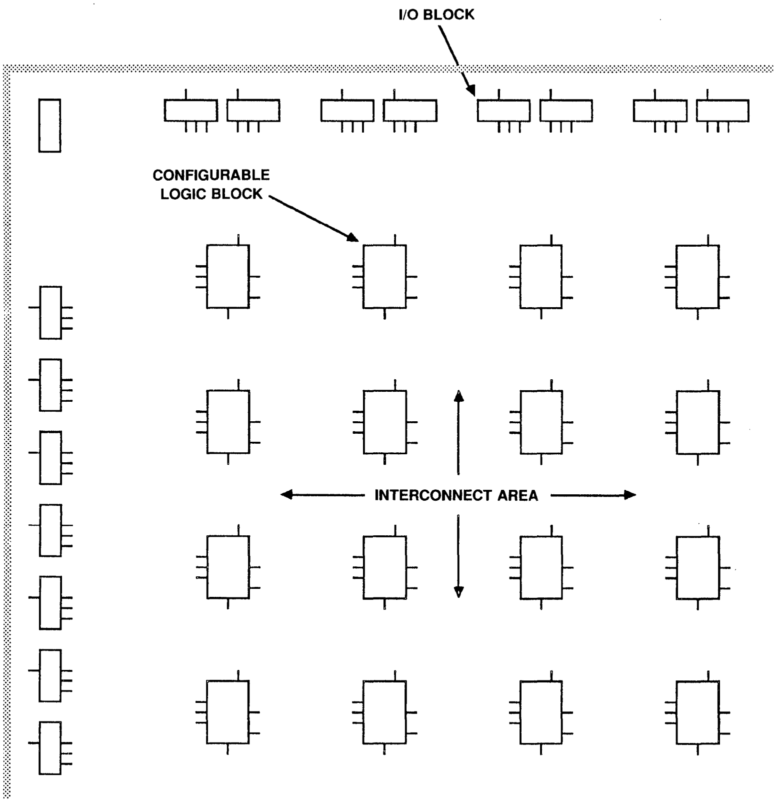


Figure 1. Logic Cell Array Structure

Configuration Memory

The configuration of the Advanced Micro Devices' Logic Cell Array is established by programming memory cells which determine the logic functions and interconnections. The memory loading process is independent of the user logic functions.

The static memory cell used for the configuration memory in the Logic Cell Array has been designed specifically for high reliability and noise immunity. Based on this design, integrity of the LCA configuration memory is assured even under adverse conditions. Compared with other programming alternatives, static memory provides the best combination of high density, high performance, high reliability and comprehensive testability. As shown in Figure 2, the basic memory cell consists of two CMOS inverters plus a pass transistor used for writing data to the cell. The cell is only written during configuration and only read during readback. During normal operation the pass transistor is "off" and does not affect the stability of the cell. This is quite different from the normal operation of conventional memory devices, in which the cells are continuously read and written.

The outputs Q and $\overline{Q}$ control pass-transistor gates directly. The absence of sense amplifiers and the output capacitive load provide additional stability to the cell. Due to the structure of the configuration memory cells, they are not affected by extreme power supply excursions or very high levels of alpha particle radiation. In reliability testing no soft errors have been observed, even in the presence of very high doses of alpha radiation.

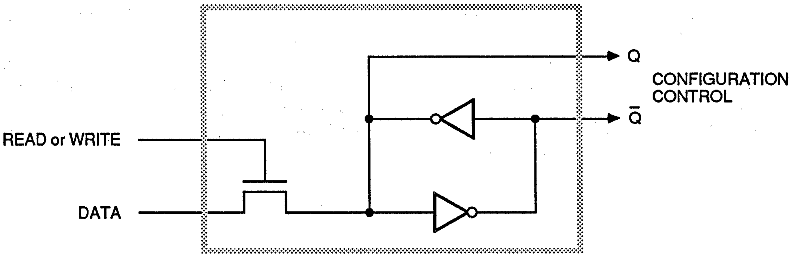


Figure 2. Configuration Memory Cell

Input/Output Block

Each user-configurable I/O block (IOB) provides an interface between the external package pin of the device and the internal logic. Each I/O block includes a programmable input path and a programmable output buffer. It also provides input clamping diodes to provide protection from electrostatic damage, and circuits to protect the LCA from latch-up due to input currents. Figure 3 shows the general structure of the I/O block.

The input buffer portion of each I/O block provides threshold detection to translate external signals applied to the package pin to internal logic levels. The input buffer threshold of the I/O blocks can be programmed to be compatible with either TTL (1.4 V) or CMOS (2.2 V) levels. The buffered input signal drives both the data input of an edge-triggered D-type flip-flop and one input of a two-input multiplexer. The output of the flip-flop provides the other input to the multiplexer. The user can select either the direct input path or the registered input, based on the content of the memory cell controlling the multiplexer. The I/O blocks along each edge of the die share common clocks. The flip-flops are reset during configuration as well as by the active-low chip $\overline{\text{RESET}}$ input.

Output buffers in the I/O blocks provide 4-mA drive for high fan-out CMOS or TTL-compatible signal levels. The output data (driving I/O block pin O) is the data source for the I/O block output buffer. Each I/O block output buffer is controlled by the contents of two configuration memory cells which turn the buffer ON or OFF or select logical three-state buffer control. The user may also select the output buffer three-state control (I/O block pin TS). When this I/O block output control signal is HIGH (a logic "1") the buffer is disabled and the package pin is high-impedance.

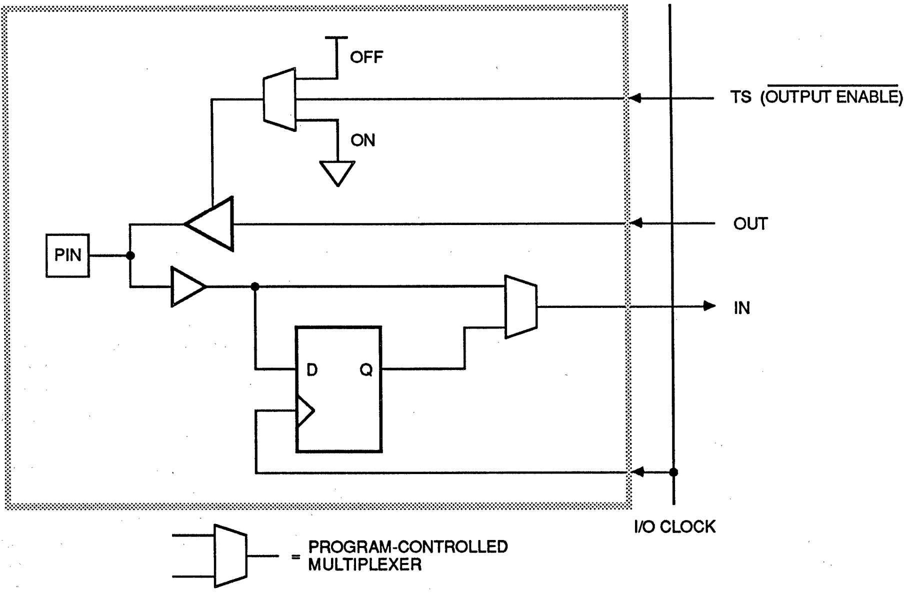

OFF
ON
TS ($\overline{\text{OUTPUT ENABLE}}$)
OUT
PIN
IN
D
Q
I/O CLOCK
= PROGRAM-CONTROLLED MULTIPLEXER

Figure 3. I/O Block

Configurable Logic Block

An Array of Configurable Logic Blocks (CLBs) provides the functional elements from which the user's logic is constructed. The logic blocks are arranged in a matrix in the center of the device. The Am2064 has sixty-four such blocks arranged in an eight-row by eight-column matrix. The Am2018 has one hundred logic blocks arranged in a ten by ten matrix. Each logic block has a combinatorial logic section, a storage element, and an internal routing and control section. Each CLB has four general-purpose inputs; A, B, C, and D; and a special clock input (K), which may be driven from the interconnect adjacent to the block. Each CLB also has two outputs, X and Y, which may drive interconnect networks. Figure 4 shows the resources of a Configurable Logic Block.

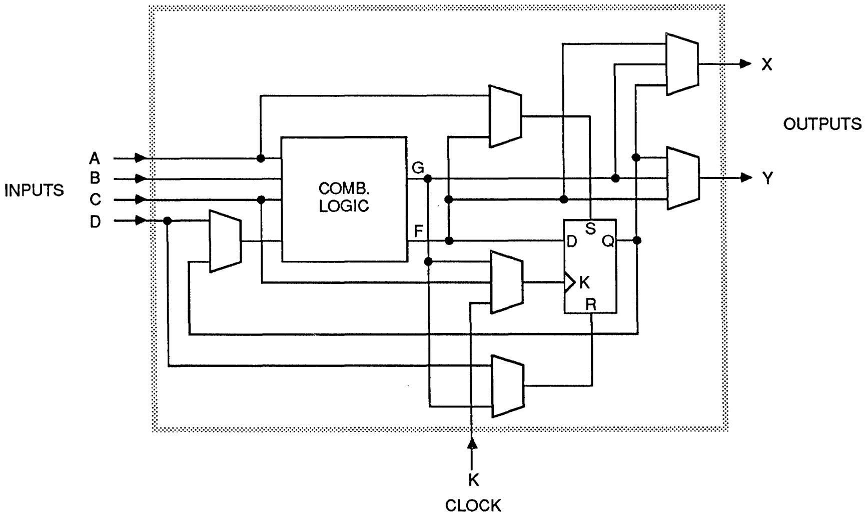


Figure 4. Configurable Logic Block

The logic block combinatorial logic uses a table look-up memory to implement Boolean functions. This technique can generate any logic function of up to four variables with a high-speed sixteen-bit memory. The propagation delay through the combinatorial network is independent of the function generated. Each block can perform any function of four variables or any two functions of three variables each. The variables may be selected from among the four inputs and the block's storage element output "Q". Figure 5 shows various options which may be specified for the combinatorial logic.

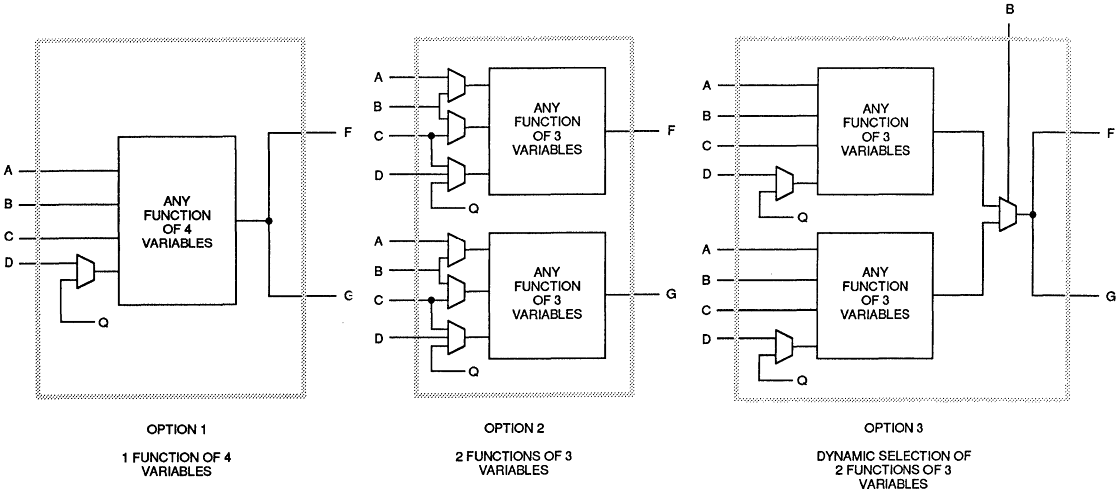

A
B
C
D
ANY FUNCTION OF 4 VARIABLES
Q
F
G
A
B
C
D
ANY FUNCTION OF 3 VARIABLES
Q
F
A
B
C
D
ANY FUNCTION OF 3 VARIABLES
Q
G
B
A
B
C
D
ANY FUNCTION OF 3 VARIABLES
Q
A
B
C
D
ANY FUNCTION OF 3 VARIABLES
Q
F
G
OPTION 1
1 FUNCTION OF 4 VARIABLES
OPTION 2
2 FUNCTIONS OF 3 VARIABLES
OPTION 3
DYNAMIC SELECTION OF 2 FUNCTIONS OF 3 VARIABLES

Figure 5. CLB Combinatorial Logic Options

If the single four-variable configuration is selected (Option 1), the F and G outputs are identical. If the two-function alternative is selected (Option 2), logic functions F and G may be independent functions of three variables each. The three variables can be selected from among the four logic block inputs and its storage element output Q. A third form of the combinatorial logic (Option 3) is a special case of the two-function form in which the B input dynamically selects between the two function tables providing a single merged logic function output. This dynamic selection allows some five-variable functions to be generated from the four block inputs and storage element Q. Combinatorial functions are restricted in that one may not use both its storage element output Q and the input variable of the logic block pin D in the same function.

If used, the storage element in each Configurable Logic Block (Figure 6) can be programmed to be either an edge-sensitive D-type flip-flop or a level-sensitive D latch. The clock or enable for each storage element can be selected from:

- The special-purpose clock input K
- The general-purpose input C
- The combinatorial function G

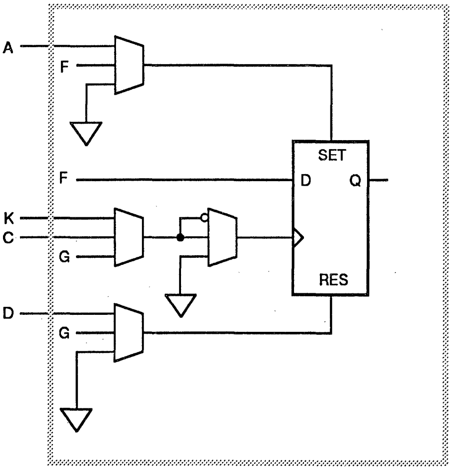


Figure 6. CLB Storage Element

The user may also select the clock active sense within each logic block. This programmable inversion eliminates the need to route both phases of a clock signal throughout the device.

The storage element data input is supplied from the function F output of the combinatorial logic. Asynchronous SET and RESET controls are provided for each storage element. The user may enable these controls independently and select their source. They are active-high inputs and the asynchronous reset is dominant. The storage elements are reset by the active-low chip $\overline{\text{RESET}}$ pin as well as by the initialization phase preceding configuration. If the storage element is not used, it is disabled.

The two block outputs, X and Y, can be driven by either the combinatorial functions, F or G, or the storage element output Q (Figure 4). Selection of the outputs is completely interchangeable and may be made to optimize routing efficiencies of the networks interconnecting the logic blocks and I/O blocks.

Programmable Interconnect

Programmable Interconnection resources in the Logic Cell Array provide routing paths to connect inputs and outputs of the I/O and logic blocks into desired networks. All interconnections are composed of metal segments, with programmable switching points provided to implement the necessary routing. Three types of resources accommodate different types of networks:

- General purpose interconnect
- Long lines
- Direct connection

General-Purpose Interconnect

General-purpose interconnect, as shown in Figure 7a, is composed of four horizontal metal segments between the rows and five vertical metal segments between the columns of logic and I/O blocks. Each segment is only the "height" or "width" of a logic block. Where these segments would cross at the intersections of rows and columns, switching matrices are provided to allow interconnections of metal segments from the adjoining rows and columns. Switches in the switch matrices and on block outputs are specially designed transistors, each controlled by a configuration bit.

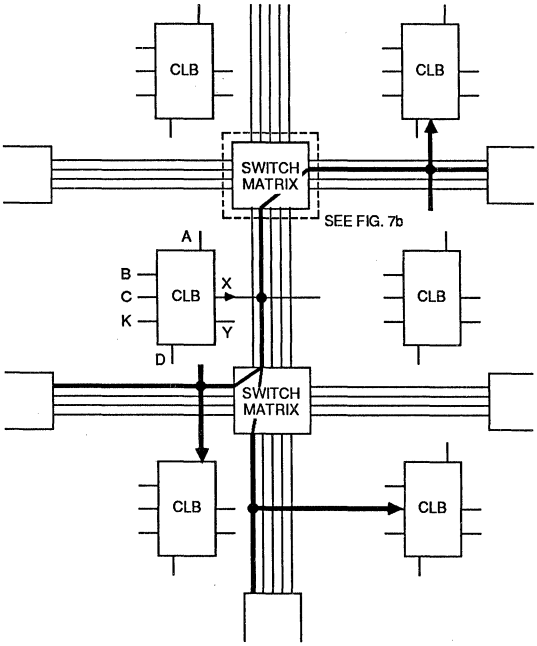
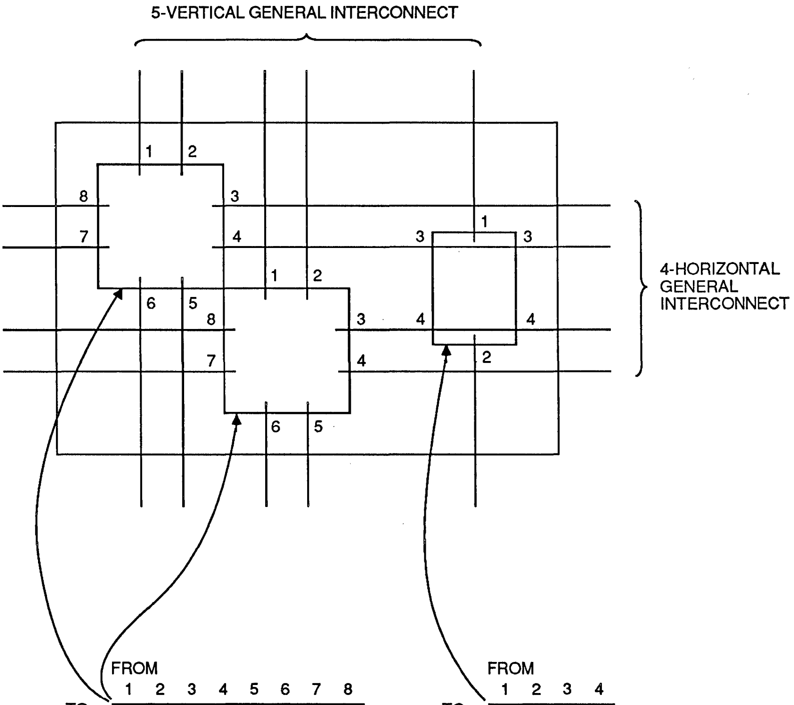


Figure 7a. General-Purpose Interconnect

Logic block output switches provide contacts to adjacent general interconnect segments and therefore to the switching matrix at each end of those segments. A switch matrix can connect an interconnect segment to other segments to form a network. Figure 7a shows the general interconnect used to route a signal from one logic block to three other logic blocks. As shown, combinations of closed switches in a switch matrix allow multiple branches for each network. The inputs of the logic or I/O blocks are multiplexers that can be programmed with configuration bits to select an input network from the adjacent interconnect segments. Since the switch connections to block inputs are unidirectional (as are block outputs) they are usable *only* for input connections. The development system software provides automatic routing of these interconnections. Interactive routing is also available for design optimization. This is accomplished by selecting a network and then toggling the states of the interconnect points by selecting them with the "mouse". In this mode, the connections through the switch matrix may be established by selecting pairs of matrix pins. The switching matrix combinations are indicated in Figure 7b.

Special buffers within the interconnect area provide periodic signal isolation and restoration for higher general interconnect fan-out and better performance. The repowering buffers are bidirectional, since signals must be able to propagate in either direction on a general interconnect segment. Direction controls are automatically established by the Logic Cell Array development system software. Repowering buffers are provided only for the general-purpose interconnect since the direct and long-line resources do not exhibit the same R-C delay accumulation. The Logic Cell Array is divided into nine sections with buffers automatically provided for general interconnect at the boundaries of these sections. These boundaries can be viewed with the development system. For routing within a section, no buffers are used. The delay calculator of the PGA Development System automatically calculates and displays the block, interconnect, and buffer delays for any selected paths.



5-VERTICAL GENERAL INTERCONNECT
1
2
8
3
7
4
6
5
1
2
8
3
7
4
6
5
1
3
3
4
4
2
4-HORIZONTAL GENERAL INTERCONNECT

FROM

TO	1	2	3	4	5	6	7	8
1	■	0	1	0	1	1	1	1
2	0	■	1	1	1	1	0	1
3	1	1	■	0	1	0	1	1
4	0	1	0	■	1	1	1	1
5	1	1	1	1	■	0	1	0
6	1	1	0	1	0	■	1	1
7	1	0	1	1	1	1	■	0
8	1	1	1	1	0	1	0	■

FROM

TO	1	2	3	4
1	■	1	1	1
2	1	■	1	1
3	1	1	■	0
4	1	1	0	■

1 = VALID CONNECTION
0 = INVALID CONNECTION

Figure 7b. Interconnection Switching Matrix

Long Lines

Long lines, shown in Figure 8a, run both vertically and horizontally the height or width of the interconnect area. Each vertical interconnection column has two long lines; each horizontal row has one, with an additional long line adjacent to each set of I/O blocks. The long lines bypass the switch matrices and are intended primarily for signals that must travel a long distance or must have minimum skew among multiple destinations.

A global buffer in the Logic Cell Array is available to drive a single signal to all B and K inputs of logic blocks. Using the global buffer for a clock provides a very low skew, high fan-out synchronized clock for use at any or all of the logic blocks. At each block, a configuration bit for the K input to the block can select this global line as the storage element clock signal. Alternatively, other clock sources can be used.

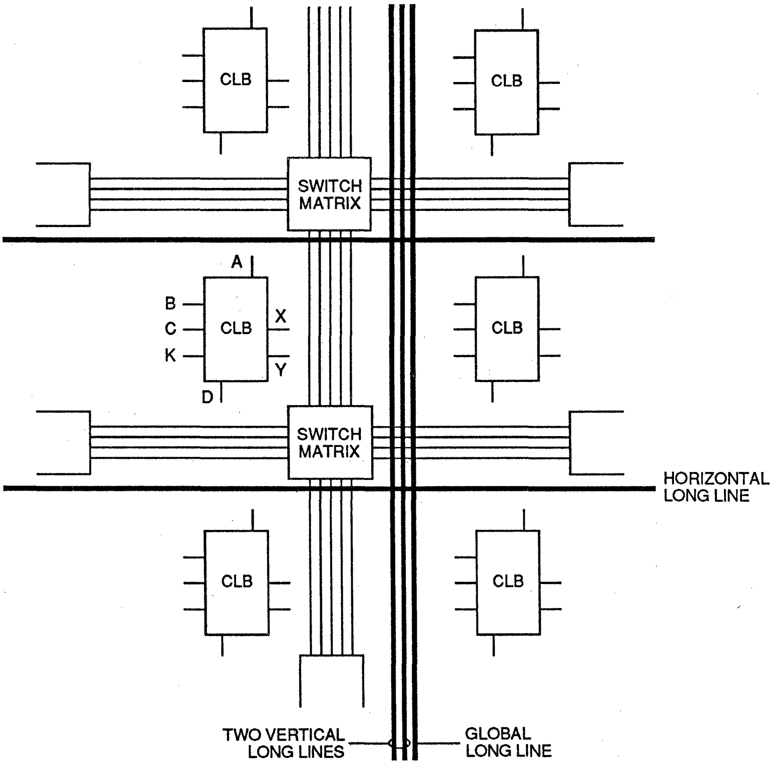


Figure 8a. Long Line Interconnect

A second buffer below the bottom row of the array drives a horizontal long line which, in turn, can drive a vertical long line in each interconnection column. This alternate buffer also has low skew and high fan-out capability. The network formed by this alternate buffer's long lines can be selected to drive the B, C or K inputs of the logic blocks. Alternatively, these long lines can be driven by a logic or I/O block on a column-by-column basis. This capability provides a common, low-skew clock or control line within each column of logic blocks. Interconnections of these long lines are shown in Figure 8b.

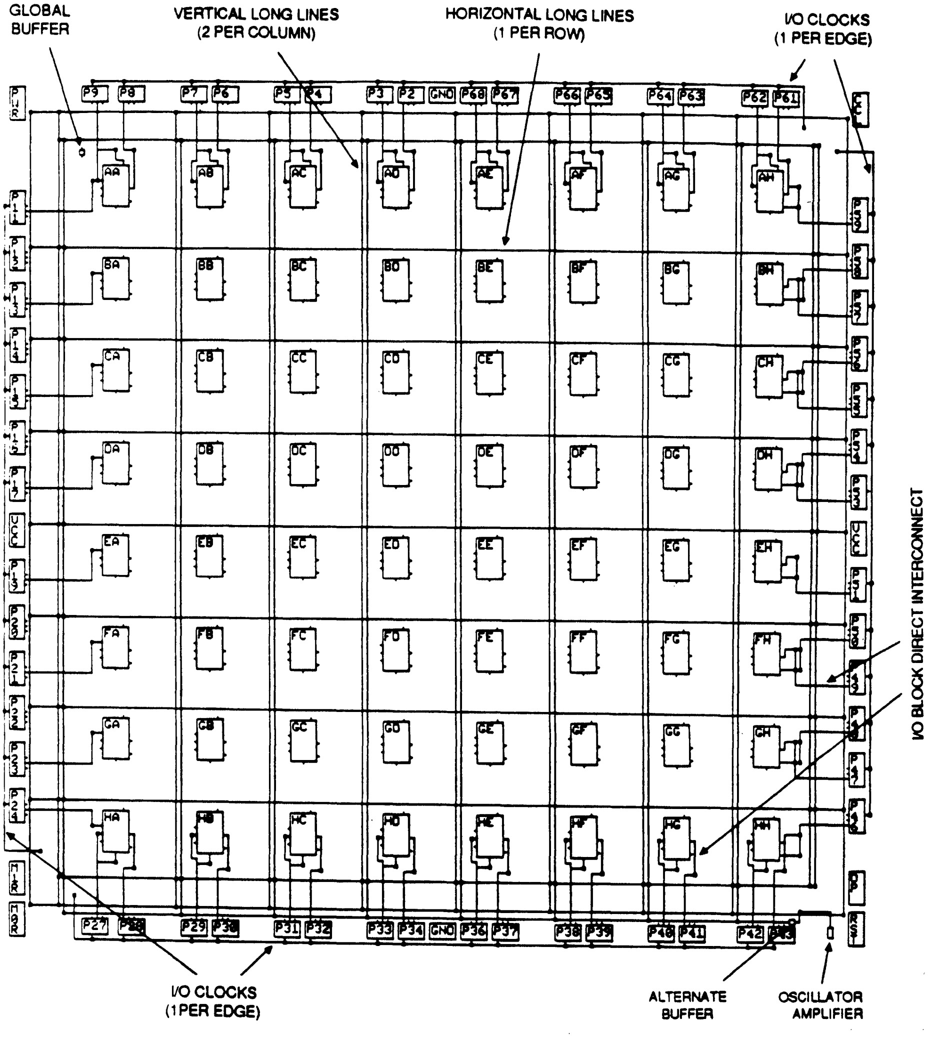


Figure 8b. Am2064 Long Lines, I/O Clocks, I/O Direct Interconnect

Direct Interconnect

Direct interconnect, shown in Figure 9, provides the most efficient implementation of networks between adjacent logic or I/O blocks. Signals routed from block to block by means of direct interconnect exhibit minimum interconnect propagation and use minimum interconnect resources. For each CLB, the X output may be connected directly to the C or D inputs of the CLB above and to the A or B inputs of the CLB below it. The Y output can use direct interconnect to drive the B input of the block immediately to its right. Where logic blocks are adjacent to I/O blocks, direct connect is provided to the I/O block input (I) on the left edge of the die, the output (O) on the right edge, or both on I/O blocks at the top and bottom of the die. Direct interconnections of I/O blocks with CLBs are shown in Figure 8b.

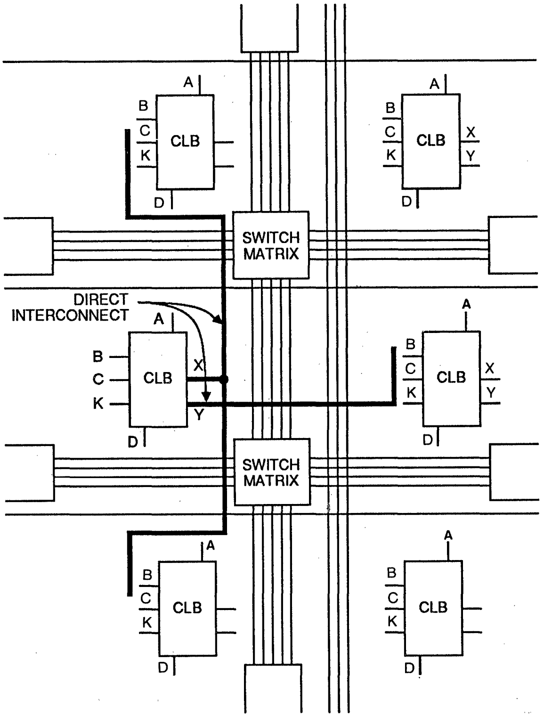
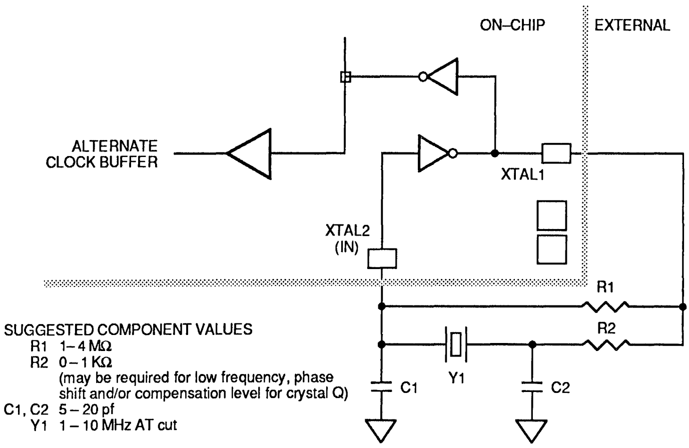


Figure 9. Direct Interconnect

Crystal Oscillator

An internal high-speed inverting amplifier is available to implement an on-chip crystal oscillator. It is associated with the auxiliary clock buffer in the lower right corner of the die. When configured to drive the auxiliary clock buffer, two special adjacent user I/O blocks are also configured to connect the oscillator amplifier with external crystal oscillator components, as shown in Figure 10. This circuit becomes active before configuration is complete in order to allow the oscillator to stabilize. Actual internal connection is delayed until completion of configuration. The feedback resistor R1 between output and input, biases the amplifier at threshold. It should be as large a value as practical to minimize loading of the crystal. The inversion of the amplifier, together with the R-C networks and crystal, produces the 360-degree phase shift of the Pierce oscillator.

A series resistor R2 may be included to add to the amplifier output impedance when needed for phase-shift control or crystal resistance matching or to limit the amplifier input swing to control clipping at large amplitudes. Excess feedback voltage may be adjusted by the ratio of C2/C1. The amplifier is designed to be used over the range from 1 MHz up to one-half the specified CLB toggle frequency. Use at frequencies below 1 MHz may require individual characterization with respect to a series resistance. Operation at frequencies above 20 MHz generally requires a crystal to operate in a third overtone mode, in which the fundamental frequency must be suppressed by the R-C networks. When the amplifier does not drive the auxiliary buffer, these I/O blocks and their package pins are available for general user I/O.



	XTAL1	XTAL2
28 PLCC	20	17
44 PLCC	29	26
48 DIP	33	30
68 PLCC	46	43
68 PGA	J10	L10
84 PLCC	56	53
84 PGA	K11	L11

Figure 10. Crystal Oscillator

Power

Power Distribution

Power for the LCA is distributed through a grid to achieve high noise immunity and isolation between logic and I/O. For packages having more than forty-eight pins, two V_{CC} pins and two ground pins are provided (see Figure 11). Inside the LCA, a dedicated V_{CC} and ground ring surrounding the logic array provides power to the I/O drivers. An independent matrix of V_{CC} and ground lines supplies the interior logic of the device. This power distribution grid provides a stable supply and ground for all internal logic, providing the external package power pins are appropriately decoupled. Typically a 0.1-µF capacitor connected between the V_{CC} and ground pins near the package will provide adequate decoupling.

Output buffers capable of driving the specified 4-mA loads under worst-case conditions may be capable of driving 25 to 30 times that current in a best case. Noise can be reduced by minimizing external load capacitance and reducing simultaneous output transitions in the same direction. It may also be beneficial to locate heavily-loaded output buffers near the ground pads. Multiple VCC and ground pin connections are required for package types which provide them.

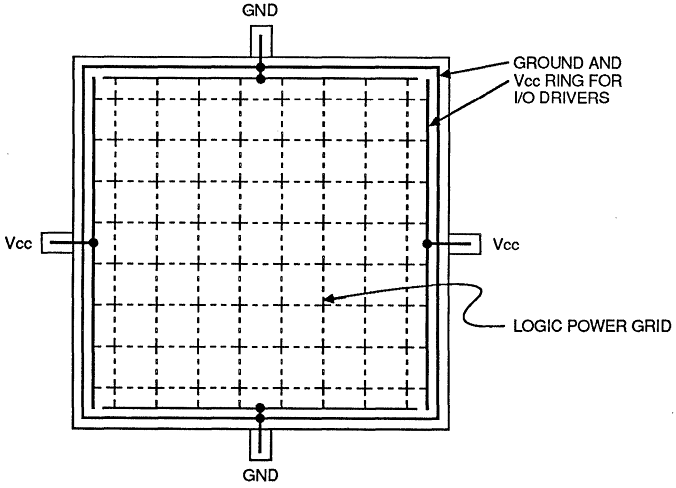


Figure 11. LCA Power Distribution

Power Dissipation

The Logic Cell Array exhibits the low power consumption characteristic of CMOS ICs. Only quiescent power is required for the LCA configured for CMOS input levels. The TTL input level configuration option requires additional power for level shifting. The power required by the static memory cells which hold the configuration data is very low and may be maintained in a power-down mode.

Typically most of power dissipation is produced by capacitive loads on the output buffers, since the power per output is 25 μW/pF/MHz. Another component of I/O power is the DC loading on each output pin. For any given system, the user can calculate the power requirement based on the resistive loading of the devices driven by the Logic Cell Array.

Internal power supply dissipation is a function of clock frequency and the number of nodes changing on each clock. In an LCA the fraction of nodes changing on a given clock is typically low (10-20%). For example, in a 16-bit binary counter, the average clock produces a change in slightly less than two of the sixteen bits. In a 4-input AND gate there will be two transitions in sixteen states. Typical global clock buffer power is about 3 mW/MHz for the Am2064 and 4 mW/MHz for the Am2018. With a "typical" load of three general interconnect segments, each CLB output requires about 0.4 mW/MHz of its output frequency. Graphs of power versus operating frequency are shown in Table 2.

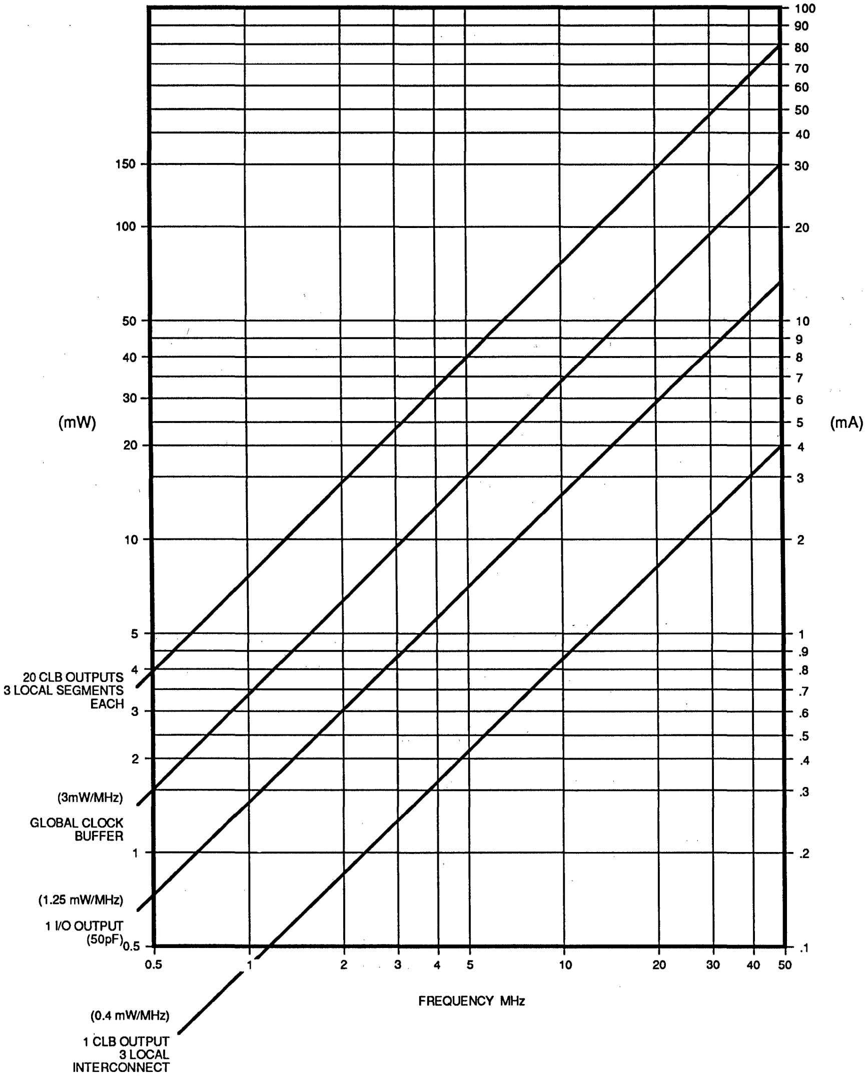
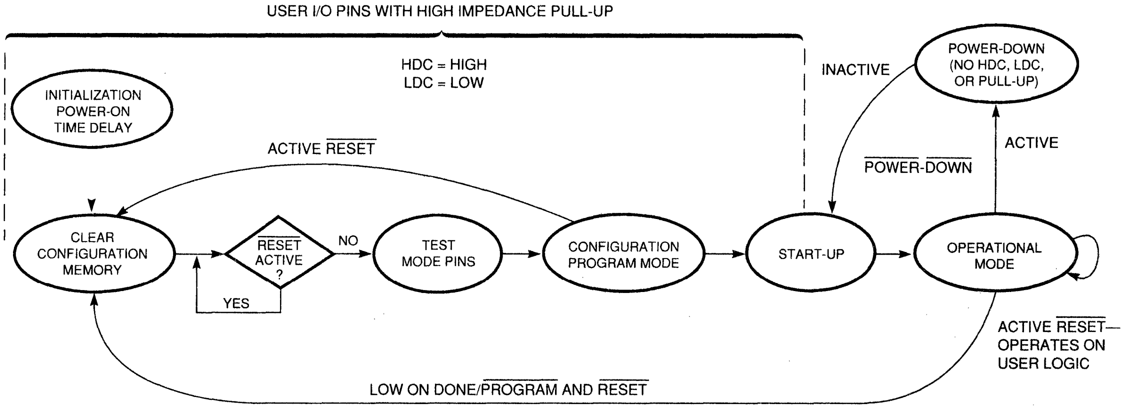


Table 2. Typical LCA Power Consumption by Element

PROGRAMMING

Configuration data to define the function and interconnection within a Logic Cell Array are loaded automatically at power-up or upon command. Several methods of automatically loading the required data are designed into the Logic Cell Array and are determined by logic levels applied to mode selection pins at configuration time. The form of the data may be either serial or parallel, depending on the configuration mode. The programming data are independent of the configuration mode selected. The state diagram of Figure 12 illustrates the configuration process.

Input thresholds for user I/O pins can be selected to be either TTL-compatible or CMOS-compatible and remain in that state until the LCA begins operation. If the user has selected CMOS compatibility, the input thresholds are changed to CMOS levels during configuration.


USER I/O PINS WITH HIGH IMPEDANCE PULL-UP
HDC = HIGH
LDC = LOW
INITIALIZATION POWER-ON TIME DELAY
ACTIVE $\overline{\text{RESET}}$
INACTIVE
POWER-DOWN (NO HDC, LDC, OR PULL-UP)
ACTIVE
$\overline{\text{POWER-DOWN}}$
CLEAR CONFIGURATION MEMORY
$\overline{\text{RESET}}$ ACTIVE ?
NO
YES
TEST MODE PINS
CONFIGURATION PROGRAM MODE
START-UP
OPERATIONAL MODE
ACTIVE $\overline{\text{RESET}}$— OPERATES ON USER LOGIC
LOW ON DONE/$\overline{\text{PROGRAM}}$ AND $\overline{\text{RESET}}$

CLEAR IS
~160 CYCLES FOR THE XC2064—100 TO 320 μs
~200 CYCLES FOR THE XC2018—125 TO 390 μs

POWER-ON DELAY IS
2^{14}CYCLES FOR THE NON-MASTER MODE—11 TO 33 ms
2^{16} CYCLES FOR THE MASTER MODE—43 TO 130 ms

1104 14

Figure 12. A State Diagram of the Configuration Process for Power-up and Reprogram

Figure 13 shows the specific data arrangement for the Am2064 device. Future Logic Cell Array products will use the same data format to maintain compatibility between different devices of AMD's product line, but they will have different sizes and numbers of data frames. For the Am2064, configuration requires 12,048 bits for each device. For the Am2018, the configuration of each device requires 17,888 bits. The Am2064 uses 160 configuration data frames and the Am2018 uses 197.

```
11111111                        DUMMY BITS (4 BITS MINIMUM)
0010                            PREAMBLE CODE
<24-BIT LENGTH COUNT>           CONFIGURATION PROGRAM LENGTH
1111                            DUMMY BITS (4 BITS MINIMUM)

0 < DATA FRAME # 001 > 111   }
0 < DATA FRAME # 002 > 111   }
0 < DATA FRAME # 003 > 111   }  Am2064:  160 CONFIGURATION DATA FRAMES
      .      .      .        }  Am2018:  197 CONFIGURATION DATA FRAMES
      .      .      .        }           (EACH FRAME CONSISTS OF:               REPEATED FOR EACH LOGIC
      .      .      .        }              A START BIT (0)                     CELL ARRAY IN A DAISY CHAIN
0 < DATA FRAME # 159 > 111   }              AN 87-BIT DATA FIELD
0 < DATA FRAME # 160 > 111   }              2 OR MORE DUMMY BITS)
1111                            POSTAMBLE CODE (4 BITS MINIMUM)
```

Figure 13. Am2064 Configuration Data Arrangement

The configuration bit stream begins with preamble bits, a preamble code and a length count. The length count is loaded into the control logic of the Logic Cell Array and is used to determine the completion of the configuration process. When configuration is initiated, a 24-bit length counter is set to 0 and begins to count the total number of configuration clock cycles applied to the device. When the current length count equals the loaded length count, the configuration process is complete. Two clocks before completion, the internal logic becomes active and is reset. On the next clock, the inputs and outputs become active as configured and consideration should be given to avoid configuration signal contention. (Attention must be paid to avoid contention on pins which are used as inputs during configuration and become outputs in operation.) On the last configuration clock, the completion of configuration is signalled by the release of the DONE/$\overline{\text{PROG}}$ pin of the device as the device begins operation. This open-drain output can be AND-tied with multiple Logic Cell Arrays and used as an active-high READY or active-low, $\overline{\text{RESET}}$, to other portions of the system. High during configuration (HDC) and low during configuration ($\overline{\text{LDC}}$), are released one CCLK cycle before DONE is asserted. In master mode configurations, it is convenient to use $\overline{\text{LDC}}$ as an active-low EPROM chip enable.

As each data bit is supplied to the LCA, it is internally assembled into a data word. As each data word is completely assembled, it is loaded in parallel into one word of the internal configuration memory array. The last word must be loaded before the current length count compare is true. If the configuration data are in error, e.g., PROM address lines swapped, the LCA will not be ready at the length count and the counter will cycle through an additonal complete count prior to configuration being "done".

Figure 14 shows the selection of the configuration mode based on the state of the mode pins M0 and M1. These package pins are sampled prior to the start of the configuration process to determine the mode to be used. Once configuration is DONE and subsequent operation has begun, the mode pins may be used to perform data readback, as discussed later. An additional mode pin, M2, must be defined at the start of configuration. This package pin is a user-configurable I/O after configuration is complete.

MODE PIN			MPDE SELECTED
M0	M1	M2	
0	0	0	Master serial
0	0	1	Master LOW mode*
0	1	1	Master HIGH mode**
1	0	1	Peripheral mode
1	1	1	Slave mode

Master LOW addresses begin at 0000 and increment.
Master HIGH addresses befin at FFFF and decrement.
*Master LOW mode not available with 28-pin package.
**Master HIGH mode not available with 28- or 44-pin package.

Figure 14. Configuration Mode Selection

Initialization Phase

When power is applied, an internal power-on-reset circuit is triggered. When VCC reaches the voltage at which the LCA begins to operate (2.5 to 3 Volts), the chip is initialized, outputs are made high-impedance and a time-out is initiated to allow time for power to stabilize. This time-out (15 to 35 ms) is determined by a counter driven by a self-generated, internal sampling clock that drives the configuration clock (CCLK) in master configuration mode. This internal sampling clock will vary with process, temperature and power supply over the range of 0.5 to 1.5 MHz. LCAs with mode lines set for master mode will time-out of their initialization using a longer counter (60 to 140 ms) to assure that all devices, which it may be driving in a daisy chain, will be ready. Configuration using peripheral or slave modes must be delayed long enough for this initialization to be completed.

The initialization phase may be extended by asserting the active-low external $\overline{\text{RESET}}$. If a configuration has begun, an assertion of $\overline{\text{RESET}}$ will initiate an abort, including an orderly clearing of partially loaded configuration memory bits. After about three clock cycles for synchronization, initialization will require about 160 additional cycles of the internal sampling clock (197 for the Am2018) to clear the internal memory before another configuration may begin. The same is true of a configured part in which the reconfigurable control bit is set. When a HIGH-to-LOW transition on the DONE/$\overline{\text{PROG}}$ package pin is detected, thereby initiating a reprogram, the configuration memory is cleared. This insures an orderly configuration in which no internal signal conflicts are generated during the loading process.

Master Mode

In master mode, the Logic Cell Array automatically loads the configuration program from an external memory device. Figure 15a shows an example of the master mode connections required. The Logic Cell Array provides sixteen address outputs and the control signals $\overline{\text{RCLK}}$ (read clock), HDC (high during configuration) and $\overline{\text{LDC}}$ (low during configuration) to execute read cycles from the external memory. Parallel eight-bit data words are read and internally serialized. As each data word is read, the least significant bit of each byte, normally D0, is the next bit in the serial stream.

Addresses supplied by the Logic Cell Array can be selected by the mode lines to begin at address 0 and incremented to read the memory (master low mode), or they can begin at address FFFF Hex and be decremented (master high mode). This capability is provided to allow the Logic Cell Array to share external memory with another device, such as a microprocessor. For example, if the processor begins its execution from low memory, the Logic Cell array can load itself from high memory and enable the processor to begin execution once configuration is completed. The DONE/$\overline{\text{PROG}}$ output pin can be used to hold the processor in a Reset state until the Logic Cell Array has completed the configuration process.

The master serial mode uses serial configuration data, synchronized by the rising edge of CCLK, as in Figure 15b.

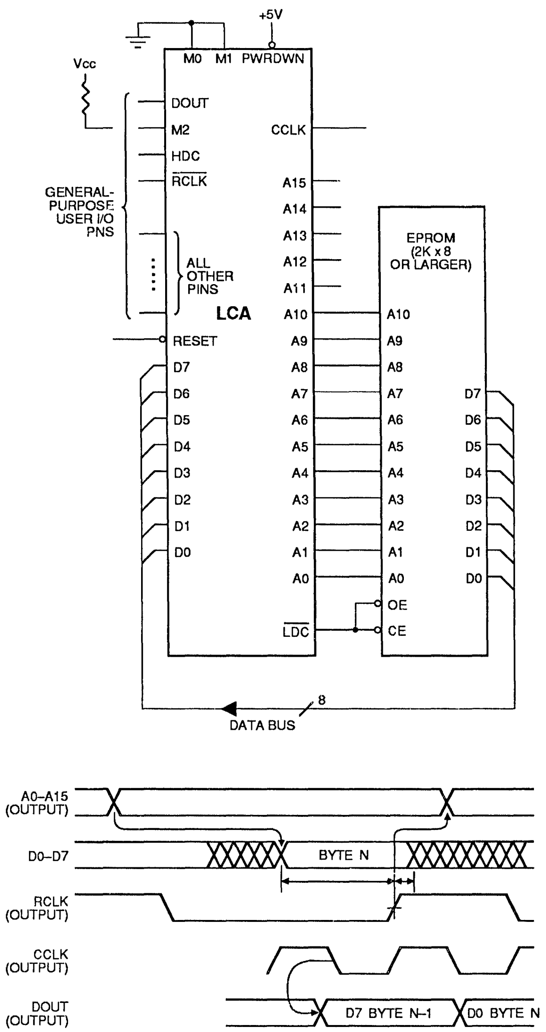


Figure 15a. Master Low Address Configuration

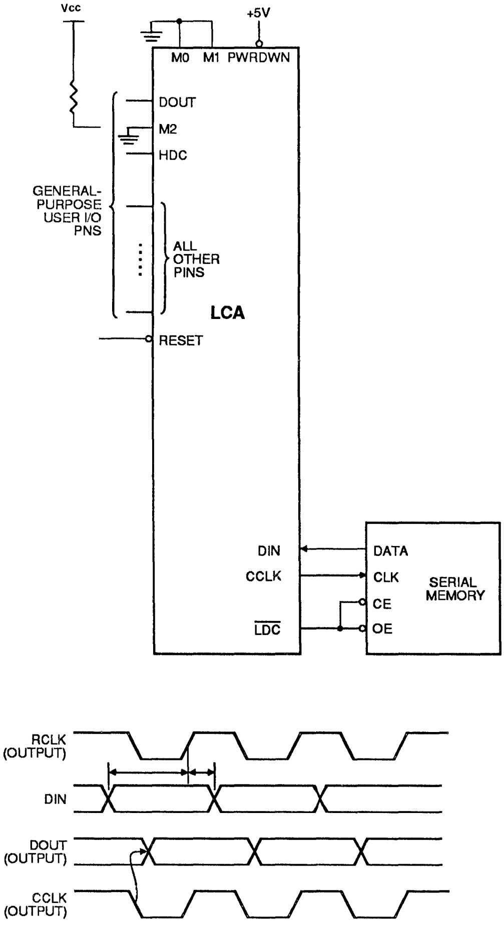


Figure 15b. Master Serial Mode Configuration

Peripheral Mode

Peripheral mode provides a simplified interface through which the device may be loaded as a processor peripheral. Figure 16 shows the peripheral mode connections. Processor write cycles are decoded from the common assertion of the active-low write strobe (WRT), and two active-low and one active-high chip selects (CS0, CS1, CS2). If all these signals are not available, the unused inputs should be driven to their respective active levels. The Logic Cell Array will accept one bit of the configuration program on the data input (DIN) pin for each processor write cycle. Data is supplied in the serial sequence described earlier.

Since only a single bit from the processor data bus is loaded per cycle, the loading process involves the processor reading a byte or word of data, writing a bit of the data to the Logic Cell Array, shifting the word and writing a bit until all bits of the word are written, then continuing in the same fashion with the next word, etc. After the configuration program has been loaded, an additional three clocks (a total of three more than the length count) must be supplied in order to complete the configuration process. When more than one device is being used in the system, each device can be assigned a different bit in the processor data bus, and multiple devices can be loaded on each processor write cycle. This "broadside" loading method provides a very easy and time-efficient method of loading several devices.

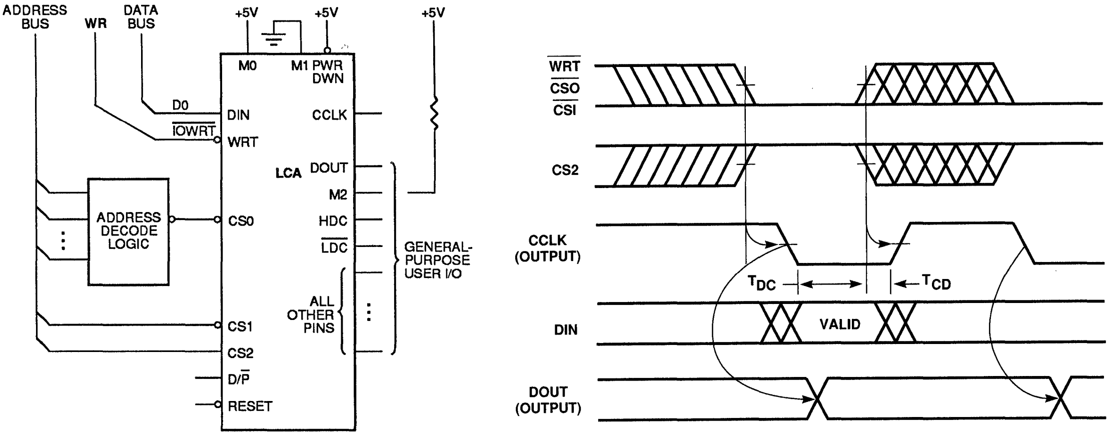


Figure 16. Peripheral Mode Configuration

Slave Mode

Slave mode, Figure 17, provides the simplest interface for loading the Logic Cell Array configuration. Data is supplied in conjunction with a synchronizing clock. For each LOW-to-HIGH input transition of configuration clock (CCLK), the data present on the data input (DIN) pin is loaded into the internal shift register. Data may be supplied by a processor or by other special circuits. Slave mode is used for downstream devices in a daisy-chain configuration. The data for each slave LCA are supplied by the preceding LCA in the chain, and the clock is supplied by the lead device, which is configured in master or peripheral mode. After the configuration program has been loaded, an additional three clocks (a total of three more than the length count) must be supplied in order to complete the configuration process.

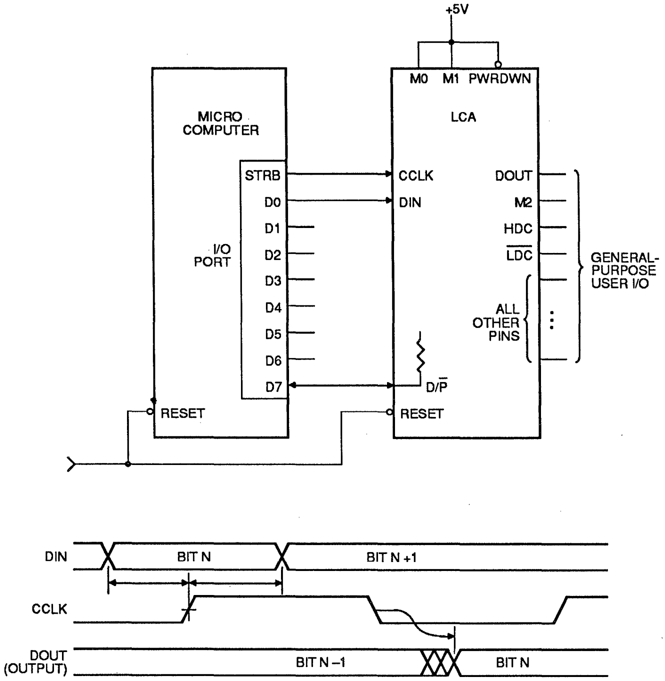


Figure 17. Slave Mode Configuration

Daisy Chain

The daisy-chain programming mode is supported by Logic Cell Array in all programming modes. In master mode and peripheral mode, the LCA can act as a source of data and control for slave devices. For example, Figure 18 shows a single device in master mode, with two devices in slave mode. The master mode device reads the external memory and begins the configuration loading process for all of the devices.

The data begin with a preamble and a length count which is supplied to all devices at the beginning of the configuration. The length count represents the total number of cycles required to load all of the devices in the daisy chain. After loading the length count, the lead device will load its configuration data while providing a HIGH DOUT to downstream devices. When the lead device has been loaded and the current length count has not reached the full value, memory access continues. Data bytes are read and serialized by the lead device. The data are passed through the lead device and appear on the data out (DOUT) pin in serial form. The lead device also generates the configuration clock (CCLK) to synchronize the serial output data. A master mode device generates an internal CCLK of eight times the EPROM address rate, while a peripheral mode device produces CCLK from the chip select and write strobe timing.

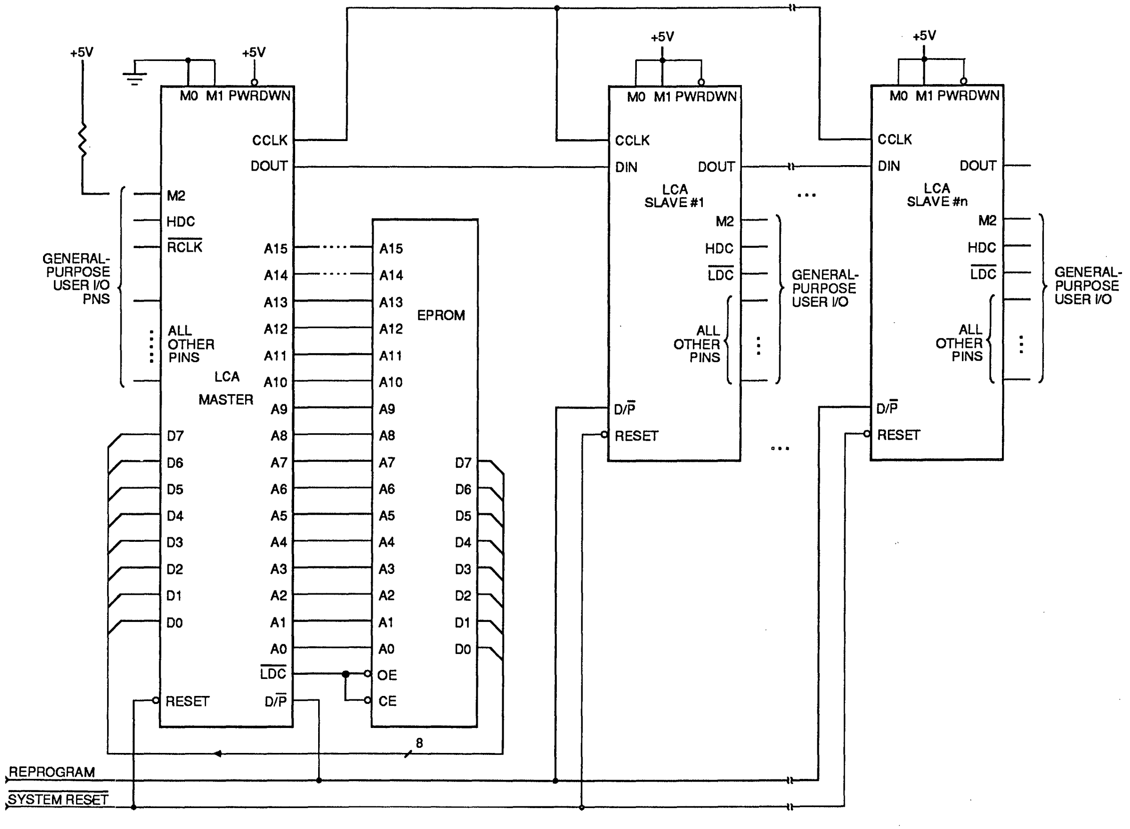


Figure 18. Master Mode with Daisy Chain

Operation

When all of the devices have been loaded and the length count is complete, a synchronous start-up of operation is performed. On the clock cycle following the end of loading, the internal logic begins functioning in the reset state. On the next CCLK, the configured output buffers become active to allow signals to stabilize. The next CCLK cycle produces the DONE condition. The length count control of operation allows a system of multiple Logic Cell Arrays to begin operation in a synchronized fashion. If the crystal oscillator is used, it will begin operation before configuration is complete to allow time for stabilization before it is connected to the internal circuitry.

SPECIAL FEATURES

In addition to the normal user logic functions and interconnect, the configuration data include control for several special functions:

- Input thresholds
- Readback enable
- Reprogram enable
- DONE pull-up resistor

Each of these functions is controlled by a portion of the configuration program generated by the PGA Development System.

Input Thresholds

During configuration, all input thresholds are TTL level. During configuration input thresholds are established as specified, either TTL or CMOS. The $\overline{\text{PWRDN}}$ input threshold is an exception; it is always a CMOS level input. The TTL threshold option requires additional power for threshold shifting.

Readback

After a Logic Cell Array has been programmed, the configuration program may be read back from the device. Readback may be used for verification of configuration and as a method of determining the state of internal logic nodes during debugging. In applications in which the verification is not used, it may be desirable to limit access to the configuration data. Three readback options are provided: 'on command', 'only once', and 'never'. If 'on-command readback' is selected, the device will respond to all readback requests. If 'readback once' is selected, the device will respond only to the first readback request after programming is complete. Subsequent readback requests will be ignored. If 'readback never' is selected, the device will not respond to a readback command.

Readback is accomplished without the use of any of the user I/O pins; only M0, M1, and CCLK pins are used. An initiation of readback is produced by a LOW-to-HIGH transition of the M0, RTRIG (read trigger) pin. Once the readback command has been given, CCLK is cycled to read back each data bit in a format similar to loading. After two dummy bits, the first data frame is shifted out, in inverted sense, on the M1, $\overline{\text{RDATA}}$ (read data) pin. All data frames must be read back to complete the process and return the mode select and CCLK pins to their normal functions.

In addition to the configuration program, the readback includes the current state of each of the internal logic block storage elements, and the state of the input (I) connection pin on each I/O block. This state information is used by the PGA Development System In-Circuit Emulator to provide visibility into the internal operation of the logic while the system is operating. To readback a uniform time sample of all storage elements it may be necessary to inhibit the system clock.

Reprogram

The configuration memory of the LCA device may be rewritten while the device is in the user's system, if that option is selected when the LCA is configured. The LCA returns to the Clear state where the configuration memory is cleared, I/O pins disabled, and mode lines resampled. Reprogram control is often implemented using an external open collector driver that pulls DONE/$\overline{\text{PROGRAM}}$ LOW. Once it recognizes a stable request, the LCA device will hold DONE/$\overline{\text{PROGRAM}}$ LOW until the new configuration has been completed. Even if the DONE/$\overline{\text{PROGRAM}}$ pin is externally held LOW beyond the configuration period, the LCA begins operation upon completion of configuration. Sensitivity to noise is reduced by confirming the HIGH-to-LOW transition over two to three cycles using the LCA's internal generator (2 to 6 µs). The Clear timeout for a Master mode reprogram or abort does not have the four times delay of the initialization state. If a daisy chain is used, an external $\overline{\text{RESET}}$ is required, long enough to guarantee clearing all non-master mode devices. This is accomplished with an external time delay.

In some applications the system power supply might have momentary failures that can leave the LCA's control logic in an invalid state. There are two methods to recover from this state. The first is to cycle the V_{CC} supply to less than 0.1 V and reapply valid V_{CC}. The second is to provide the LCA with simultaneous LOW levels of at least 6 µs on $\overline{\text{RESET}}$ and DONE/$\overline{\text{PROGRAM}}$ pins after the $\overline{\text{RESET}}$ pin has been HIGH following a return to valid V_{CC}. This guarantees that the LCA returns to the Clear state. Either of these methods may be used in the event of an incomplete voltage interruption. They are not needed for a normal application of power from an off condition.

DONE Pull-Up

The DONE/$\overline{\text{PROG}}$ pin is an open drain I/O that indicates programming status. As an input it initiates a reprogram operation. An optional internal pull-up resistor may be enabled.

Battery Backup

Because the control store of the Logic Cell Array is a CMOS static memory, its cells require only a very low standby current for data retention. In some systems, this low data retention current characteristic facilitates preserving configurations in the event of a primary power loss. The Logic Cell Array has built in power-down logic which, when activated, will disable normal operation of the device and retain only the configuration data. All internal operation is suspended and output buffers are placed in their high-impedance state.

Power-down data retention is possible with a simple battery-backup circuit because the power requirement is extremely low. For retention at 2.0 V, the required current is typically on the order of 0.5 mA. Screening of this parameter is available. To force the Logic Cell Array into the power-down state, the user must pull the $\overline{\text{PWRDWN}}$ pin low and continue to supply a retention voltage to the V_{CC} pins of the package. When normal power is restored, V_{CC} is elevated to its normal operating voltage and $\overline{\text{PWRDWN}}$ is returned to a HIGH. The Logic Cell Array resumes operation with the same internal sequence that occurs at the conclusion of configuration. Internal I/O and logic block storage elements will be reset, the outputs will become enabled and then the DONE/$\overline{\text{PROG}}$ pin will be released. No configuration programming is involved.

PERFORMANCE

The high performance of the Logic Cell Array results from its patented architectural features and from the use of an advanced high-speed CMOS manufacturing process. Performance may be measured in terms of minimum propagation times for logic elements.

Flip-flop loop delays for the I/O block and logic block flip-flops are about 3 ns. This short delay provides very good performance under asynchronous clock and data conditions. Short loop delays minimize the probability of a metastable condition which can result from assertion of the clock during data transitions. Because of the short loop delay characteristic in the Logic Cell Array, the I/O block flip-flops can be used very effectively to synchronize external signals applied to the device. Once synchronized in the I/O block, the signals can be used internally without further consideration of their clock relative timing, except as it applies to the internal logic and routing path delays.

Device Performance

The single parameter which most accurately describes the overall performance of the Logic Cell Array is the maximum toggle rate for a logic block storage element configured as a toggle flip-flop. The configuration for determining the toggle performance of the Logic Cell Array is shown in Figure 19. The clock for the storage element is provided by the global clock buffer and the flip-flop output Q is fed back through the combinatorial logic to form the data input for the next clock edge. Using this arrangement, flip-flops in the Logic Cell Array can be toggled at clock rates from 33-100 MHz, depending on the speed grade used.

Actual Logic Cell Array performance is determined by the critical path speed, including both the speed of the logic and storage elements in that path, and the speed of the particular network routing. Figure 20 shows a typical system logic configuration of two flip-flops with an extra combinatorial level between them. Depending on speed grade, system clock rates to 35 MHz are practical for this logic. To allow the user to make the best use of the capabilities of the device, the delay calculator in the PGA Development System determines worst-case path delays using actual impedance and loading information.

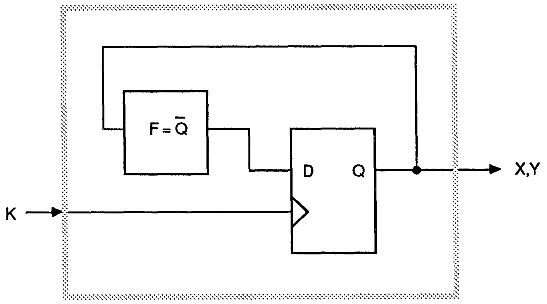


Figure 19. Logic Block Configuration for Toggle Rate Measurement

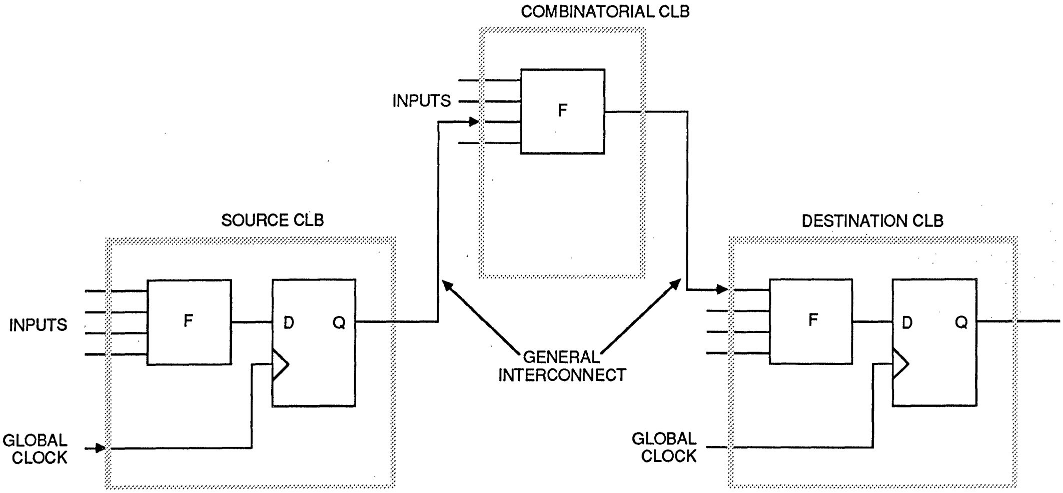
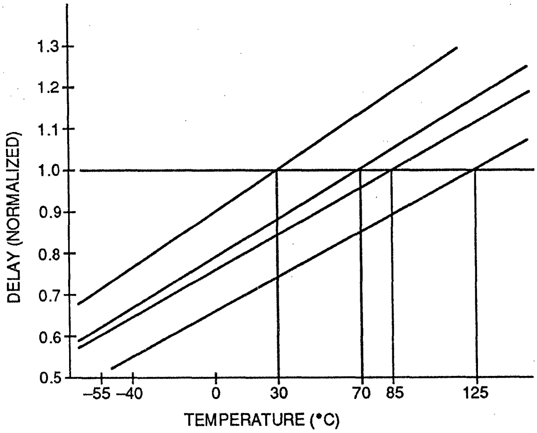


Figure 20. Typical Logic Path

Logic Block Performance

Logic Block propagation times are measured from the interconnect point at the input of the combinatorial logic to the output of the block in the interconnect area. Combinatorial performance is independent of logic function because of the table look-up based implementation. Timing is different when the combinatorial logic is used in conjunction with the storage element. For the combinatorial logic function driving the data input of the storage element, the critical timing is data set-up relative to the clock edge provided to the storage element. The delay from the clock source to the output of the logic block is critical in the timing of signals produced by storage elements. The loading on a logic block output is limited only by the additional propagation delay of the interconnect network. Performance of the logic block is a function of supply voltage and temperature, as shown in Figures 21 and 22.



NOTE: NORMALIZED FOR FOUR TEMPERATURES

Figure 21. Delay vs. Temperature

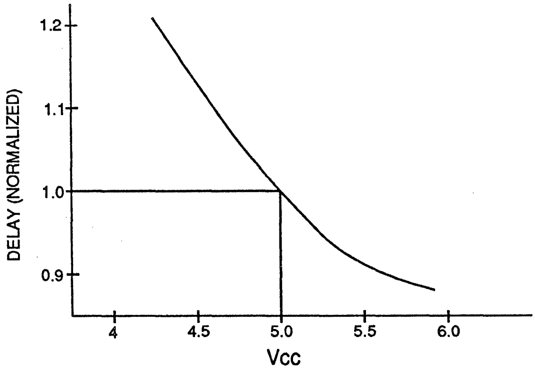


Figure 22. Delay vs. Power Supply

Interconnect Performance

Interconnect performance depends on the routing resource used to implement the signal path. As discussed earlier, direct interconnect from block to block provides a minimum delay path for a signal.

The single metal segment used for long lines exhibits low resistance from end to end, but relatively high capacitance. Signals driven through a programmable switch will have the additional impedance of the switch added to their normal drive impedance.

General-purpose interconnect performance depends on the number of switches and segments used, the presence of the bidirectional repowering buffers and the overall loading on the signal path at all points along the path. In calculating the worst-case delay for a general interconnect path, the delay calculator portion of the PGA Development System accounts for all of these elements. As an approximation, interconnect delay is proportional to the summation of totals of local metal segments beyond each programmable switch. In effect, the delay is a sum of R-C delays each approximated by an R times the total C it drives. The R of the switch and the C of the interconnect are functions of the particular device performance grade. For a string of three local interconnects, the approximate delay at the first segment, after the first switch resistance, would be three units; an additional two delay units after the next switch plus an additional delay after the last switch in the chain. The interconnect R-C chain terminates at each repowering buffer. Nearly all of the capacitance is in the interconnect metal and switches; the capacitance of the block inputs is not significant. Figure 23 shows an estimation of this delay.

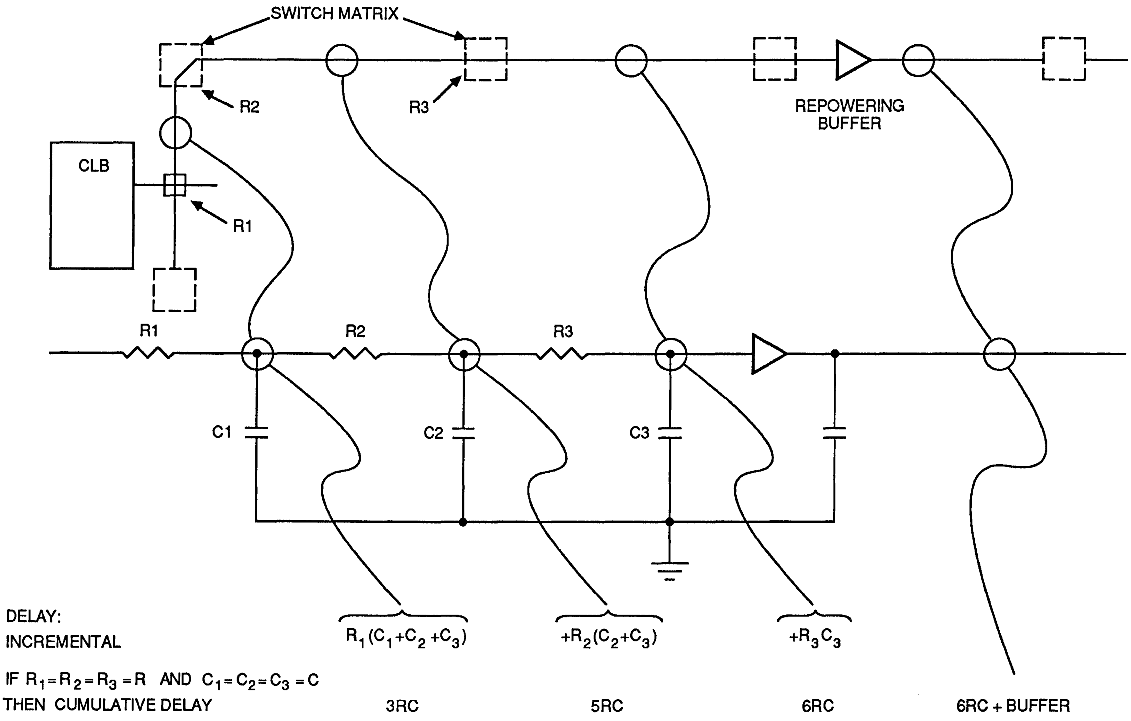


Figure 23. Interconnection Delay Example

28-PIN PLCC	48-PIN DIP	68-PIN PLCC	68-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
				SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
		1	B6	GND				
		2	A6	≪HIGH≫		A13 (O)		I/O
	1	3	B5			A6 (O)		
		4	A5			A12 (O)		
2	2	5	B4			A7 (O)		
	3	6	A4			A11 (O)		
3	4	7	B3			A8 (O)		
	5	8	A3			A10 (O)		
4	6	9	A2			A9 (O)		
5	7	10	B2	$\overline{\text{PWRDWN}}$ (I)				
6	8	11	B1	≪HIGH≫				I/O
		12	C2					
	9	13	C1					
		14	D2					
7	10	15	D1					
		16	E2					
	11	17	E1					
8	12	18	F2	VCC				
	13	19	F1	≪HIGH≫				I/O
		20	G2					
9	14	21	G1					
		22	H2					
	15	23	H1					
	16	24	J2					
10	17	25	J1	M1 (HIGH)	M1 (LOW)	M1 (HIGH)	M1 (LOW)	$\overline{\text{RDATA}}$ (O)
11	18	26	K1	M0 (HIGH)	M0 (HIGH)	M0 (LOW)	M0 (LOW)	RTRIG (I)
12	19	27	K2	M2 (HIGH)				I/O
13	20	28	L2	HDC (HIGH)				
		29	K3	≪HIGH≫				
14	21	30	L3	LDC (LOW)				
		31	K4	≪HIGH≫				
	22	32	L4					
		33	K5					
	23	34	L5					

≪HIGH≫ is high impedance with a 20- to 50-KΩ internal pull-up resistor during configuration

Table 2a. Am2064 Pin Assignments (Continued on next page)

<table>
<tr><th rowspan="2">28-PIN PLCC</th><th rowspan="2">48-PIN DIP</th><th rowspan="2">68-PIN PLCC</th><th rowspan="2">68-PIN PGA</th><th colspan="4">CONFIGURATION MODE: ≪M2: M1: M0≫</th><th rowspan="2">USER OPERATION</th></tr>
<tr><th>SLAVE
<1:1:1></th><th>PERIPHERAL
<1:0:1></th><th>MASTER-HIGH
<1:1:0></th><th>MASTER-LOW
<1:0:0></th></tr>
<tr><td>15</td><td>24</td><td>35</td><td>K6</td><td colspan="4">GND</td><td></td></tr>
<tr><td></td><td></td><td>36</td><td>L6</td><td colspan="4" rowspan="5">≪HIGH≫</td><td rowspan="7">I/O</td></tr>
<tr><td>16</td><td>25</td><td>37</td><td>K7</td></tr>
<tr><td></td><td></td><td>38</td><td>L7</td></tr>
<tr><td></td><td>26</td><td>39</td><td>K8</td></tr>
<tr><td></td><td>27</td><td>40</td><td>L8</td></tr>
<tr><td></td><td>28</td><td>41</td><td>K9</td><td colspan="2" rowspan="3"></td><td colspan="2">D7 (I)</td></tr>
<tr><td></td><td>29</td><td>42</td><td>L9</td><td colspan="2">D6 (I)</td></tr>
<tr><td>17</td><td>30</td><td>43</td><td>L10</td><td colspan="2"></td><td>XTL2 or I/O</td></tr>
<tr><td>18</td><td>31</td><td>44</td><td>K10</td><td colspan="5">$\overline{\text{RESET}}$ (I)</td></tr>
<tr><td>19</td><td>32</td><td>45</td><td>K11</td><td colspan="4">DONE (O)</td><td>$\overline{\text{PROG}}$ (I)</td></tr>
<tr><td>20</td><td>33</td><td>46</td><td>J10</td><td colspan="4" rowspan="2">≪HIGH≫</td><td>XTL1 or I/O</td></tr>
<tr><td></td><td></td><td>47</td><td>J11</td><td rowspan="5">I/O</td></tr>
<tr><td></td><td>34</td><td>48</td><td>H10</td><td colspan="2" rowspan="2"></td><td colspan="2">D5 (I)</td></tr>
<tr><td></td><td></td><td>49</td><td>H11</td><td colspan="2"></td></tr>
<tr><td>21</td><td>35</td><td>50</td><td>G10</td><td rowspan="2"></td><td>$\overline{\text{CS0}}$ (I)</td><td colspan="2">D4 (I)</td></tr>
<tr><td>22</td><td>36</td><td>51</td><td>G11</td><td>$\overline{\text{CS1}}$ (I)</td><td colspan="2">D3 (I)</td></tr>
<tr><td></td><td></td><td>52</td><td>F10</td><td colspan="5">VCC</td></tr>
<tr><td></td><td></td><td>53</td><td>F11</td><td rowspan="5">≪HIGH≫</td><td colspan="3"></td><td rowspan="7">I/O</td></tr>
<tr><td></td><td>37</td><td>54</td><td>E10</td><td>CS2 (I)</td><td colspan="2">D2 (I)</td></tr>
<tr><td></td><td></td><td>55</td><td>E11</td><td colspan="3"></td></tr>
<tr><td>23</td><td>38</td><td>56</td><td>D10</td><td>$\overline{\text{WRT}}$ (I)</td><td colspan="2">D1 (I)</td></tr>
<tr><td></td><td>39</td><td>57</td><td>D11</td><td></td><td colspan="2">$\overline{\text{RCLK}}$</td></tr>
<tr><td>24</td><td>40</td><td>58</td><td>C10</td><td colspan="2">DIN (I)</td><td colspan="2">D0 (I)</td></tr>
<tr><td>25</td><td>41</td><td>59</td><td>C11</td><td colspan="4">DOUT (O)</td></tr>
<tr><td>26</td><td>42</td><td>60</td><td>B11</td><td>CCLK (I)</td><td colspan="3">CCLK (O)</td><td>CCLK (I)</td></tr>
<tr><td></td><td>43</td><td>61</td><td>B10</td><td colspan="2" rowspan="8">≪HIGH≫</td><td colspan="2">A0 (O)</td><td rowspan="8">I/O</td></tr>
<tr><td>27</td><td>44</td><td>62</td><td>A10</td><td colspan="2">A1 (O)</td></tr>
<tr><td></td><td>45</td><td>63</td><td>B9</td><td colspan="2">A2 (O)</td></tr>
<tr><td>28</td><td>46</td><td>64</td><td>A9</td><td colspan="2">A3 (O)</td></tr>
<tr><td></td><td></td><td>65</td><td>B8</td><td colspan="2">A15 (O)</td></tr>
<tr><td></td><td>47</td><td>66</td><td>A8</td><td colspan="2">A4 (O)</td></tr>
<tr><td></td><td></td><td>67</td><td>B7</td><td colspan="2">A14 (O)</td></tr>
<tr><td>1</td><td>48</td><td>68</td><td>A7</td><td colspan="2">A5 (O)</td></tr>
</table>

≪HIGH≫ is high impedance with a 20- to 50-KΩ internal pull-up resistor during configuration

Table 2a. Am2064 Pin Assignments (Continued)

44-PIN PLCC	68-PIN PLCC	68-PIN PGA	84-PIN PLCC	84-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
					SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
1	1	B6	1	C6	GND				
	2	A6	2	A6	≪HIGH≫		A13 (O)		I/O
			3	A5					
			4	B5					
2	3	B5	5	C5			A6 (O)		
	4	A5	6	A4			A12 (O)		
3	5	B4	7	B4			A7 (O)		
4	6	A4	8	A3			A11 (O)		
5	7	B3	9	A2			A8 (O)		
6	8	A3	10	B3			A10 (O)		
7	9	A2	11	A1			A9 (O)		
8	10	B2	12	B2	$\overline{\text{PWRDWN}}$ (I)				
9	11	B1	13	C2	≪HIGH≫				I/O
	12	C2	14	B1					
10	13	C1	15	C1					
	14	D2	16	D2					
11	15	D1	17	D1					
			18	E3					
	16	E2	19	E2					
			20	E1					
	17	E1	21	F2					
12	18	F2	22	F3	VCC				
	19	F1	23	G3	≪HIGH≫				I/O
			24	G1					
13	20	G2	25	G2					
			26	F1					
14	21	G1	27	H1					
	22	H2	28	H2					
15	23	H1	29	J1					
	24	J2	30	K1					
16	25	J1	31	J2	M1 (HIGH)	M1 (LOW)	M1 (HIGH)	M1 (LOW)	RDATA (O)
17	26	K1	32	L1	M0 (HIGH)	M0 (HIGH)	M0 (LOW)	M0 (LOW)	RTRIG (I)
18	27	K2	33	K2	M2 (HIGH)				I/O
19	28	L2	34	K3	HDC (HIGH)				
	29	K3	35	L2	≪HIGH≫				
20	30	L3	36	L3	LDC (LOW)				
	31	K4	37	K4	≪HIGH≫				
21	32	L4	38	L4					
			39	J5					
	33	K5	40	K5					
	34	L5	41	L5					
22			42	K6					

≪HIGH≫ is high impedance with a 20- to 50-KΩ internal pull-up resistor during configuration

Table 2b. Am2018 Pin Assignments (Continued on next page)

44-PIN PLCC	68-PIN PLCC	68-PIN PGA	84-PIN PLCC	84-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
					SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
23	35	K6	43	J6	GND				
			44	J7	≪HIGH≫				I/O
	36	L6	45	L7					
	37	K7	46	K7					
	38	L7	47	L6					
			48	L8					
	39	K8	49	K8					
	40	L8	50	L9					
24	41	K9	51	L10			D7 (I)		
25	42	L9	52	K9			D6 (I)		
26	43	L10	53	L11					XT2 or I/O
27	44	K10	54	K10	$\overline{\text{RESET}}$ (I)				
28	45	K11	55	J10	DONE (0)				$\overline{\text{PROG}}$ (1)
29	46	J10	56	K11	≪HIGH≫				XTL1 or I/O
	47	J11	57	J11					I/O
30	48	H10	58	H10			D5 (I)		
			59	H11					
	49	H11	60	F10					
			61	G10					
31	50	G10	62	G11		$\overline{\text{CS0}}$ (I)	D4 (I)		
32	51	G11	63	G9		$\overline{\text{CS1}}$ (I)	D3 (I)		
33	52	F10	64	F9	VCC				
	53	F11	65	F11	≪HIGH≫				I/O
34	54	E10	66	E11		CS2 (I)	D2 (I)		
			67	E10					
	55	E11	68	E9					
			69	D11					
35	56	D10	70	D10		$\overline{\text{WRT}}$ (I)	D1 (I)		
	57	D11	71	C11			$\overline{\text{RCLK}}$		
36	58	C10	72	B11	DIN (I)		D0 (I)		
37	59	C11	73	C10	DOUT (O)				
38	60	B11	74	A11	CCLK (I)	CCLK (O)			CCLK (I)
39	61	B10	75	B10	≪HIGH≫		A0 (O)		I/O
40	62	A10	76	B9			A1 (O)		
41	63	B9	77	A10			A2 (O)		
42	64	A9	78	A9			A3 (O)		
	65	B8	79	B8			A15 (O)		
43	66	A8	80	A8			A4 (O)		
	67	B7	81	B6			A14 (O)		
			82	B7					
			83	A7					
44	68	A7	84	C7			A5 (O)		

≪HIGH≫ is high impedance with a 20- to 50-KΩ internal pull-up resistor during configuration

Table 2b. Am2018 Pin Assignments (Continued)

STANDARD PRODUCT CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Supply voltage, V	-0.5 V to 7 V
Power down, V	3.5 V to 7 V
Input voltage	-0.5 V to V_{CC} +0.5 V
Voltage applied to three-state output	-0.5 V to V_{CC} +0.5 V
Storage temperature range	-65°C to +150°C
Lead temperature (soldering, 10 seconds)	260°C

Stresses beyond those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those listed under RECOMMENDED OPERATING CONDITIONS is not implied. Exposure to ABSOLUTE MAXIMUM RATINGS conditions for extended periods of time may affect device reliability.

OPERATING CONDITIONS

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply voltage relative to GND	Commercial	4.75		5.25	V
		Industrial	4.50		5.50	
V_{IHT}	High level input voltage–TTL configuration		2.0		V_{CC}	V
V_{IHC}	High level input voltage–CMOS configuration		0.7 V_{CC}		V_{CC}	V
V_{ILT}	Low level input voltage–TTL configuration		0		0.8	V
V_{ILC}	Low level input voltage–CMOS configuration		0		0.2 V_{CC}	V
I_{IT}	Input leakage current–TTL configuration		±10			μA
I_{IC}	Input leakage current–CMOS configuration		±10			μA
I_{OZ}	Three-state output off current (V_{CC} = 5.5 V)		±10			μA
t_{OP}	Operating free-air temperature		0		70	°C

ELECTRICAL CHARACTERISTICS Over Operating Conditions

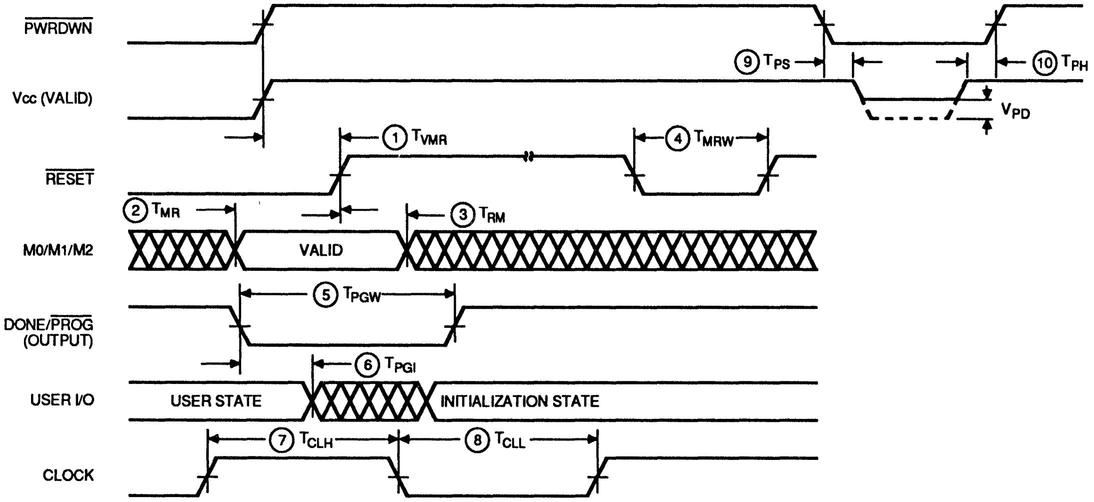
Symbol	Parameter		Test Condition		Min	Typ	Max	Unit
V_{OH}	High level output voltage	Commercial	V_{CC} = 4.75 V	I_{OH} = -4.0 mA	3.86			V
		Industrial	V_{CC} = 4.5 V	I_{OH} = -4.0 mA	3.76			
V_{OL}	Low level output voltage	Commercial	V_{CC} = 4.75 V	I_{OL} = 4.0 mA			0.32	V
		Industrial	V_{CC} = 4.5 V	I_{OL} = 4.0 mA			0.37	
I_{CCO}	Quiescent operating power supply current	CMOS inputs	V_{CC} = 5.0 V				5	mA
		TTL inputs	V_{CC} = 5.0 V				10	
I_{CCPD}	Power down supply current		V_{CC} = 5.0 V			0.5		mA

POWER ON TIMING

The LCAs contain on-chip reset timing logic for power-up operation. To insure proper master mode system operation, VCC must rise from 2.0 V to minimum specification level in 10 ms or less. For other modes, initiation of configuration must be delayed for 60 ms after VCC reaches the minimum specified level.

SWITCHING CHARACTERISTICS – GENERAL

Symbol	Description		-33		-50		-70		-100		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{VMR}①	RESET (2)	V_{CC} setup (2.0 V)	150		150		150		150		ns
t_{MR}②		M2, M1, M0 setup	60		60		60		60		ns
t_{RM}③		M2, M1, M0 hold	60		60		60		60		ns
t_{MRW}④		Width (LOW)	150		150		150		150		ns
t_{PGW}⑤	DONE/ PROG	Program width (LOW)	6		6		6		6		µs
t_{PGI}⑥		Initialization		7		7		7		7	µs
t_{CLH}⑦	CLOCK	Clock (HIGH)	12		8		7		7		ns
t_{CLL}⑧		Clock (LOW)	12		8		7		7		ns
t_{PS}⑨	$\overline{\text{PWR}}\ \overline{\text{DWN}}$	Setup to V_{CC}	0		0		0		0		ns
t_{PH}⑩		Hold from V_{CC}	0		0		0		0		ns
V_{PD}		Power Down	2.0		2.0		2.0		2.0		V

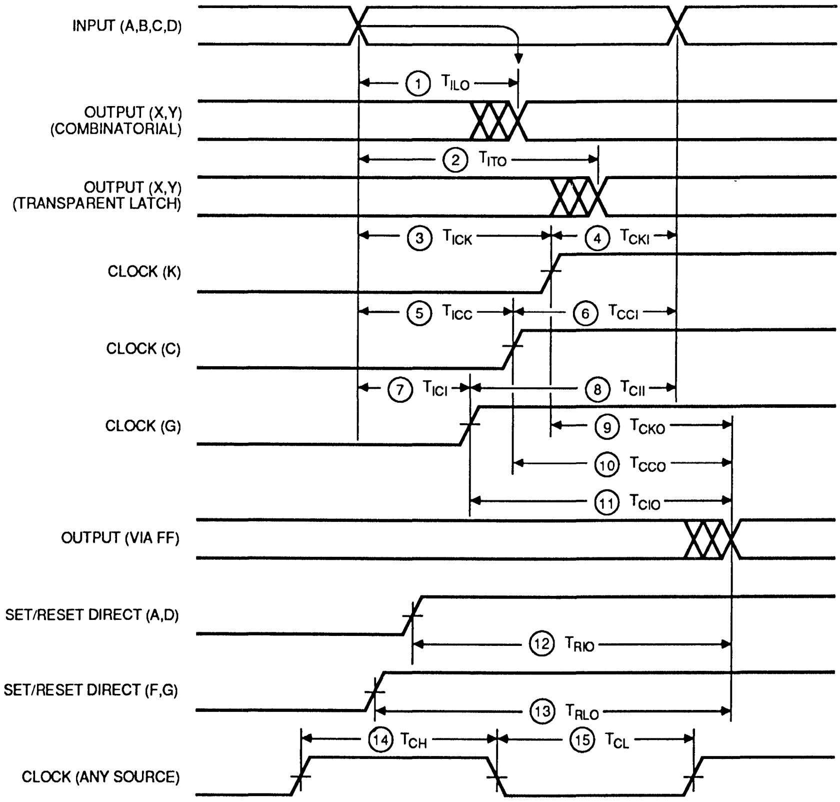


SWITCHING CHARACTERISTICS – CLB

Symbol	Description		-33		-50		-70		-100		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{ILO}①	Logic input to output	Combinatorial		20		15		10		7.0	ns
t_{ITO}②		Transparent latch		25		20		14		10	ns
t_{QLO}		Additional for Q through F or G to out		13		8		6		5	ns
t_{CKO}⑨	K Clock	To output		20		15		10		7.0	ns
t_{ICK}③		Logic-input setup	12		8		7		5		ns
t_{CKI}④		Logic-input hold	0		0		0		0		ns
t_{CCO}⑩	C Clock	To output		25		19		13		9.0	ns
t_{ICC}⑤		Logic-input setup	12		9		6		5		ns
t_{CCI}⑥		Logic-input hold	6		0		0		0		ns
t_{CIO}⑪	Logic input to G Clock	To output		37		27		20		13.0	ns
t_{ICI}⑦		Logic-input setup	6		4		3		2		ns
t_{CII}⑧		Logic-input hold	9		5		4		3		ns
t_{RIO}⑫	Set/reset direct	Input A or D to out		25		22		16		11	ns
t_{RLO}⑬		Through F or G to out		37		28		21		14	ns
t_{MRQ}		Master Reset pin to out		35		25		20		17	ns
t_{RS}		Separation of set/reset	17		9		7		5		ns
t_{RPW}		Set/reset pulse-width	12		9		7		6		ns
F_{CLK}	Flip-flop toggle rate	Q through F to flip-flop	33		50		70		100		MHz
t_{CH}⑭	Clock	Clock HIGH	12		8		7		5		ns
t_{CI}⑮		Clock LOW	12		8		7		5		ns

Note: All switching characteristics apply to all valid combinations of process, temperature and supply.

SWITCHING CHARACTERISTICS – CLB

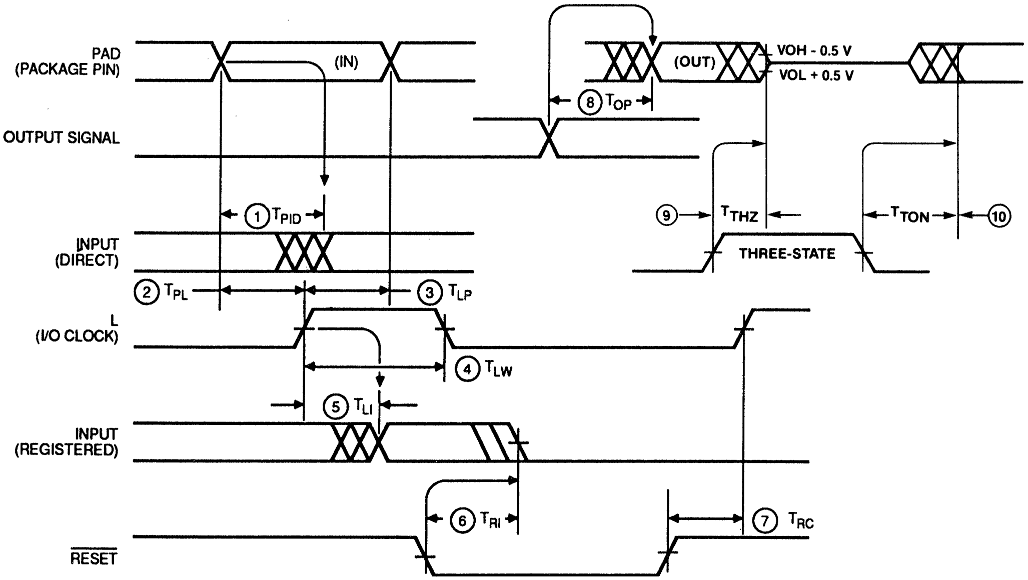


3

SWITCHING CHARACTERISTICS – IOB

Symbol	Description	-33		-50		-70		-100		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{PID}①	Pad (package pin) to input (direct)		12		8		6		5	ns
t_{LI}⑤	I/O Clock to input (storage)		20		15		11		7.0	ns
t_{PL}②	I/O Clock to pad-input setup	12		8		6		4		ns
t_{LP}③	I/O Clock to pad-input hold	0		0		0		0		ns
t_{LW}④	I/O Clock pulse width	12		9		7		5		µs
	I/O Clock frequency	33		50		70		100		MHz
t_{OP}⑧	Output to pad (output enabled)		15		12		9		7.0	ns
t_{THZ}⑨	Three-state to pad begin hi-Z		25		20		15		13	ns
t_{TON}⑩	Three-state to pad end hi-Z		25		20		15		13	ns
t_{RI}⑥	RESET to input (storage)		40		30		25		17	ns
t_{RC}⑦	RESET to input clock		35		25		20		14	ns

Note: Timing is measured at 0.5 V_{CC} levels with 50 pF output load.

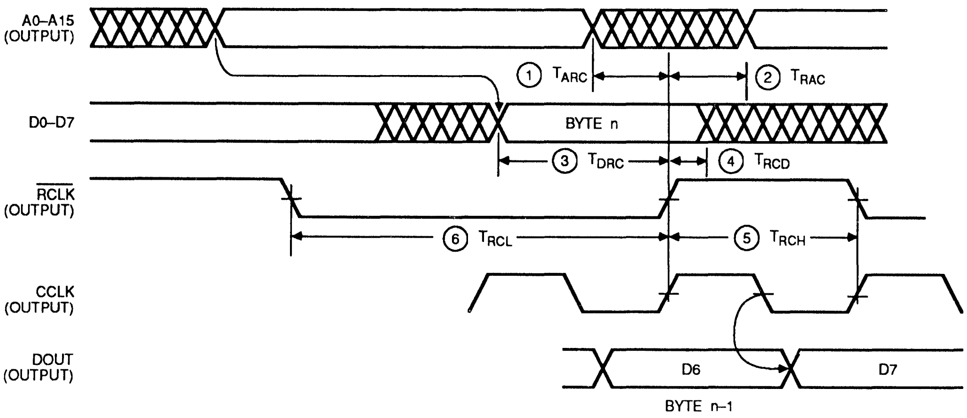


SWITCHING CHARACTERISTICS – PROGRAMMING – MASTER MODE

Symbol	Description		-33		-50		-70		-100		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{ARC}①	RCLK	From address invalid		0		0		0		0	ns
t_{RAC}②		To address valid		200		200		200		200	ns
t_{DRC}③		To data setup	60		60		60		60		ns
t_{RCD}④		To data hold	0		0		0		0		ns
t_{RCH}⑤		$\overline{\text{RCLK}}$ HIGH	600		600		600		600		ns
t_{RCL}⑥		$\overline{\text{RCLK}}$ LOW	4.0		4.0		4.0		4.0		µs

Notes: 1. CCLK and DOUT timing are the same as for slave mode.

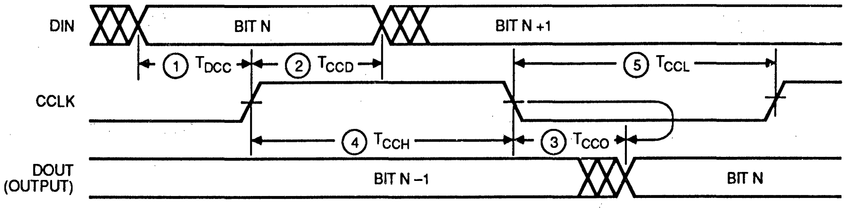
2. At power up, V_{CC} must rise from 2.0 to V_{CC} minmimum in less than 10 ms.



SWITCHING CHARACTERISTICS – PROGRAMMING – SLAVE MODE

Symbol	Description	-33		-50		-70		-100		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{CCO}③	CCLK to DOUT		65		65		65		65	ns
t_{DCC}①	CCLK DIN setup	0		0		0		0		ns
t_{CCD}②	CCLK DIN hold	40			40		40		40	ns
t_{CCH}④	CCLK HIGH time	0.25		0.25		0.25		0.25		µs
t_{CCL}⑤	CCLK LOW time	0.25	5.0	0.25	5.0	0.25	5.0	0.25	5.0	µs
F_{CC}	CCLK frequency		2		2		2		2	MHz

Note: Configuration must be delayed at least 40 ms after V_{CC} minimum.



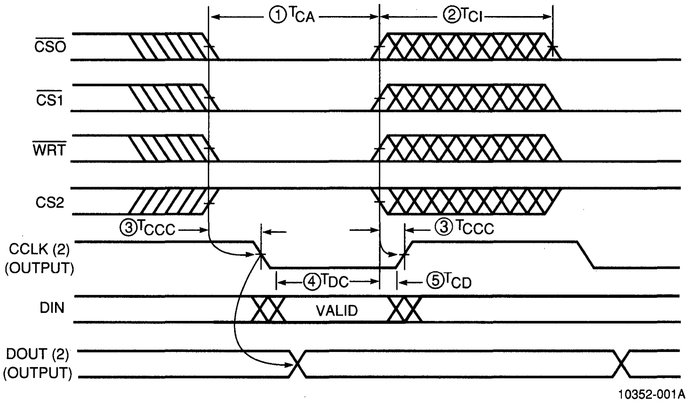
SWITCHING CHARACTERISTICS – PROGRAMMING – PERIPHERAL MODE

Symbol	Description		-33		-50		-70		-100		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{CA}①	Controls[1] ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$)	Active (last active input to first inactive)	0.25	5.0	0.25	5.0	0.25	5.0	0.25	5.0	μs
t_{CI}②		Inactive (first inactive input to last active)	0.25		0.25		0.25		0.25		μs
t_{CCC}③		CCLK[2]		75		75		75		7.5	ns
t_{DC}④		DIN setup	35		35		35		35		ns
t_{CD}⑤		DIN hold	5		5		5		5		ns

Notes: 1. Peripheral mode timing determined from last control signal of the logical AND of ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$) to transition to, active or inactive state.

2. CCLK and DOUT timing are the same as for slave mode.

3. Configuration must be delayed at least 40 ms after VCC minimum.

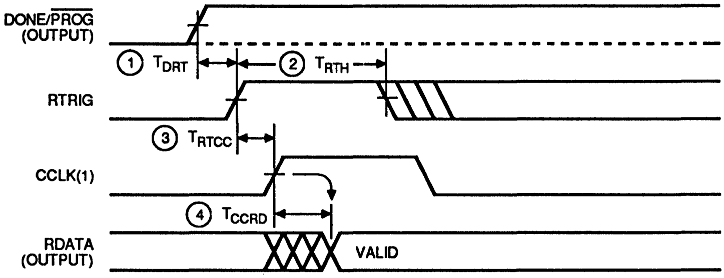


SWITCHING CHARACTERISTICS – PROGRAM READBACK

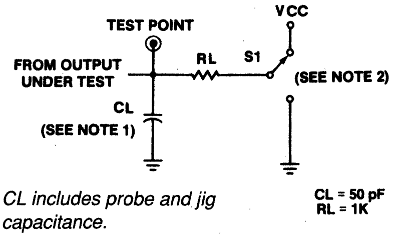
Symbol	Description		-33		-50		-70		-100		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{DRT}①	RTRIG	PROG Setup	300		300		300		300		ns
t_{RTH}②		RTRIG HIGH	250		250		250		250		ns
t_{RTCC}③	CCLK	RTRIG Setup	100		100		100		100		ns
t_{CCRD}④		RDATA Delay		100		100		100		100	ns

Notes: 1. CCLK and DOUT timing are the same as for slave mode.

2. DONE/PROG output/input must be HIGH (device programmed) prior to a positive transition of RTRIG (M0).



SWITCHING TEST LOAD



Note: CL includes probe and jig capacitance.

DESIGN AIDS

Designing with the Logic Cell Array is similar to using conventional MSI elements or gate array macrocells. The first step is to partition the desired logic design into Logic Blocks and I/O blocks, usually based on shared input variables or efficient use of flip-flop and combinatorial logic. Following a plan for placement of the blocks, the design information may be entered using the interactive Graphic Design Editor. The design information includes both the functional specifications for each block and a definition of the interconnection networks. A macrocell library provides a simplified entry of commonly-used logic functions. As an alternative to interactive block placement and configuration, a schematic may be created using elements from the macrocell library. Automatic placement and routing is available for either method of design entry. After routing the interconnections, various checking stages and processing of that data are performed to ensure that the design is correct. Design changes may be implemented in minutes. The design file is used to generate the programming data which can be downloaded directly into an LCA in the target system and operated. The program information may be used to program PROM, EPROM or ROM devices, or stored in some other media as needed by the final system.

Design verification may be accomplished by using the AMD XACTOR™ In-Circuit Emulation System directly in the target system and/or the ORCAD/VST or the P-Silos ™ logic simulator.

RECOMMENDED SOCKETS

The following sockets, with matching hole patterns, are available for PLCC devices.

	DESCRIPTION	VENDOR	PART NUMBER
68-pin	PCB solder tail, tin plate	AMP	821574-1
	Surface mount, tin plate	AMP	821542-1
	PCB solder tail, tin plate	Burndy*	QILE68P-410T
	PCB solder tail, tin plate	Midland-Ross*	709-2000-068-4-1-1
	PCB solder tail, tin plate	Methode*	213-068-001
	Surface mount, tin plate	Methode*	213-068-002
84-pin	PCB solder tail, tin plate	AMP	821573-1
	Surface mount, tin plate	AMP	821546-1
	PCB solder tail, tin plate	Burndy*	QILE84P-410T
	PCB solder tail, tin plate	Midland-Ross*	709-2000-084-4-1-1
	PCB solder tail, tin plate	Methode*	213-084-001
	Surface mount, tin plate	Methode*	213-084-002

** Sockets will plug into pin-grid array (PGA) wire-wrap sockets for breadboard use.*

MILITARY PRODUCT CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Supply voltage, V_{CC} ....-0.5 V to 7 V
Power down, V_{CC} ....3.5 V to 7 V
Input voltage ....-0.5 V to V_{CC} +0.5 V
Voltage applied to three-state output.........-0.5 V to V_{CC} +0.5 V
Storage temperature ....-65°C to +150°C
Maximum soldering temperature ....260°C

MILITARY OPERATING CONDITIONS

Symbol	Parameter	Min	Typ	Max	Unit
V_{CC}	Supply voltage relative to GND	4.5		5.5	V
V_{IHT}	High level input voltage–TTL configuration	2.0		V_{CC}	V
V_{IHC}	High level input voltage–CMOS configuration	0.7 V_{CC}		V_{CC}	V
V_{ILT}	Low level input voltage–TTL configuration	0		0.8	V
V_{ILC}	Low level input voltage–CMOS configuration	0		0.2 V_{CC}	V
I_{IT}	Input leakage current–TTL configuration	±10			µA
I_{IC}	Input leakage current–CMOS configuration	±10			µA
I_{OZ}	Three-state output off current (V_{CC} = 5.5 V)	±10			µA
T_A	Operating free-air temperature	-55			°C
T_C	Operating case temperature			+125	°C

ELECTRICAL CHARACTERISTICS Over Military Operating Conditions

SYMBOL	PARAMETER		TEST CONDITION		MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage		V_{CC} = MIN	I_{OH} = –4.0 mA	3.7			V
V_{OL}	Low level output voltage		V_{CC} = MIN	I_{OL} = 4.0 mA			0.4	V
I_{CCO}	Quiescent operating power supply current	CMOS inputs-Am2064	V_{CC} = 5.0 V				10	mA
		CMOS inputs-Am2018	V_{CC} = 5.0 V				15	mA
		TTL inputs	V_{CC} = 5.0 V				10	mA
I_{CCPD}	Power down supply current		V_{CC} = 5.0 V				0.5	mA

POWER ON TIMING

The LCAs contain on-chip reset timing logic for power-up operation. To insure proper master mode system operation, VCC must rise from 3.5 V to minimum specification level in 10 ms or less. For other modes, initiation of configuration must be delayed for 60 ms after VCC reaches the minimum specified level.

TEST CONDITIONS

Outputs loaded with rated DC current and 50-pF capacitance to GND.

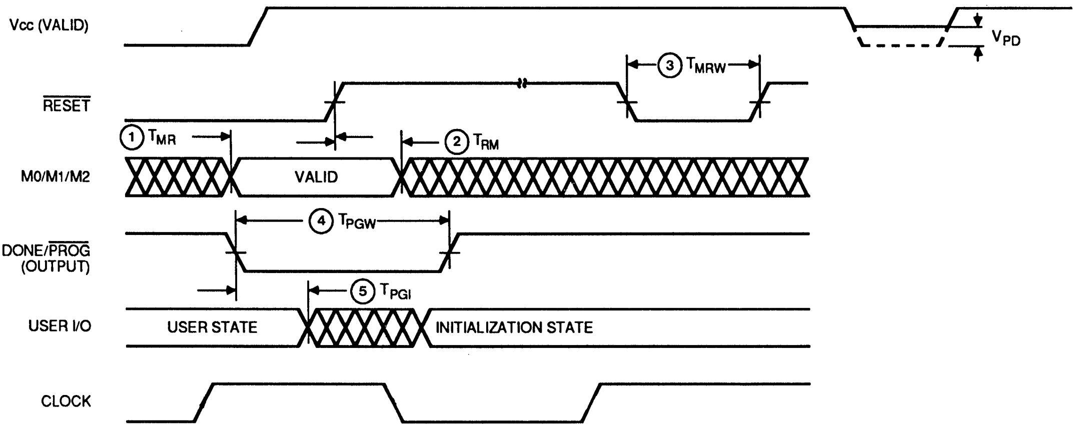
APPLICATION GUIDELINES, POWER ON SETUP CONDITIONS* – GENERAL

Symbol	Description		-20		-33, -50, -70 *		Unit
			Min	Max	Min	Max	
t_{MR}①	$\overline{\text{RESET}}$ [1]	M2, M1, M0 setup	1		1		µs
t_{RM}②		M2, M1, M0 hold	1		1		µs
t_{MRW}③		Width (LOW)	250		150		ns
t_{PGW}④	DONE/ $\overline{\text{PROG}}$	Program width (LOW)	6		6		µs
t_{PGI}⑤		Initialization		7		7	µs
V_{PD}	$\overline{\text{PWR}}$ $\overline{\text{DWN}}$	Power Down	3.5		3.5		V

* 70 MHz specifications are preliminary.

Notes: 1. $\overline{\text{RESET}}$ timing relative to power-on and valid mode lines (M0, M1, M2) is relevant only when $\overline{\text{RESET}}$ is used to delay configuration.

** Not directly tested.*



APPLICATION GUIDELINES, SWITCHING CHARACTERISTICS – CLB

Symbol	Description		-20		-33		-50		-70*		Unit
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{ILO}①	Logic input to output	Combinatorial		35		20		15		10	ns
t_{ITO}②		Transparent latch		45		25		20		14	ns
t_{QLO}		Additional for Q through F or G to out		30		13		8		6	ns
t_{CKO}⑨	K Clock	To output		35		20		15		10	ns
t_{ICK}③		Logic-input setup	22		12		8		7		ns
t_{CKI}④		Logic-input hold	1		1		1		1		ns
t_{CCO}⑩	C Clock	To output		45		25		19		13	ns
t_{ICC}⑤		Logic-input setup	18		12		9		6		ns
t_{CCI}⑥		Logic-input hold	10		6		1		1		ns
t_{CIO}⑪	Logic input to G Clock	To output		65		37		27		20	ns
t_{ICI}⑦		Logic-input setup	10		6		4		3		ns
t_{CII}⑧		Logic-input hold	15		9		5		4		ns
t_{RIO}⑫	Set/reset direct	Input A or D to out		45		25		22		16	ns
t_{RLO}⑬		Through F or G to out		65		37		28		21	ns
t_{MRQ}		Master Reset pin to out		60		55		45		20	ns
t_{RS}		Separation of set/reset	30		17		9		7		ns
t_{RPW}		Set/reset pulse-width	20		12		9		7		ns
F_{CLK}	Flip-flop toggle rate	Q through F to flip-flop	20		33		50		70		MHz
t_{CH}⑭	Clock	Clock HIGH	20		12		8		7		ns
t_{CL}⑮		Clock LOW	20		12		8		7		ns

* 70 MHz specifications are preliminary.

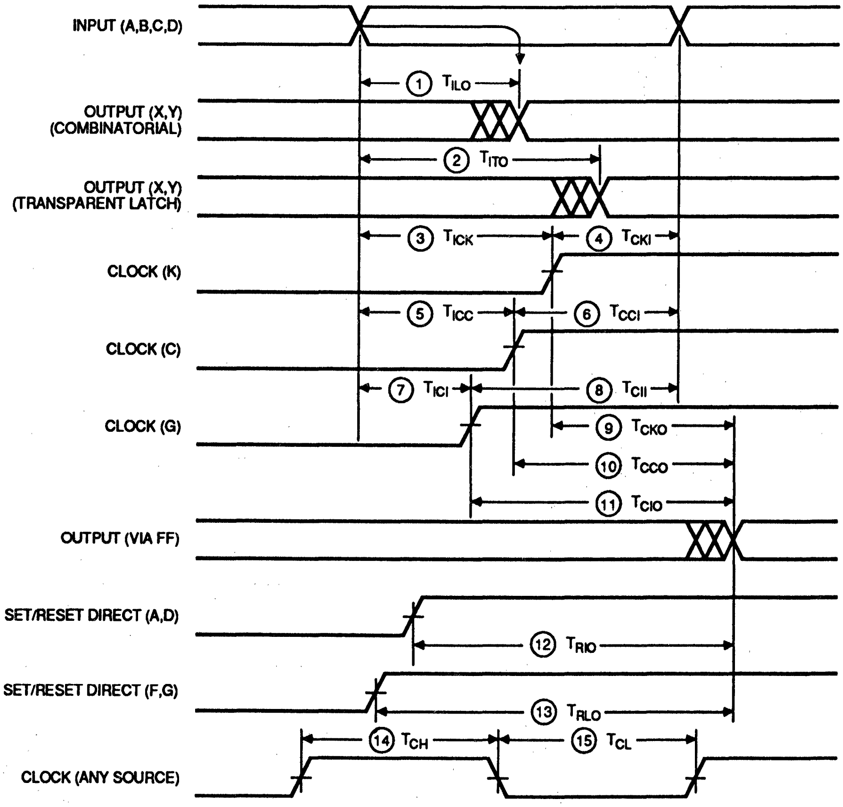
BENCHMARK PATTERNS*

Part Number	Test	Sym	Conditions -55°C ≤ T_C ≤ +125°C V_{CC} = 5.0 V ±10%	Group A Subgroups	-20 Limits		-33 Limits		-50 Limits		-70* Limits		Units
					Min	Max	Min	Max	Min	Max	Min	Max	
Am2018	T_{PID} + interconnect + 10 (T_{ILO}) + T_{OP}. Measured on 10 cols.	T_{B1}		9,10,11		410		238		178		119	ns
Am2064	T_{PID} + interconnect + 8 (T_{ILO}) + T_{OP}. Measured on 8 cols.	T_{B1}		9,10,11		345		187		140		N/A	ns

* 70 MHz specifications are preliminary.

Note: Testing of the Applications Guidelines is modeled after testing specified by MIL-M-38510/605. Devices are first 100% functionally tested. Benchmark patterns are then used to determine our compliance to the Application Guidelines. Characterization data are taken at initial device qualification, prior to introduction of significant changes, and at least twice yearly to monitor correlation between patterns, device performance, XACT software timings, and the data sheet.

SWITCHING CHARACTERISTICS CLB



MILITARY CASE OUTLINES*

Package Outline Letter	Conforms MIL-M 38510 Appendix C Case
P	P-BC

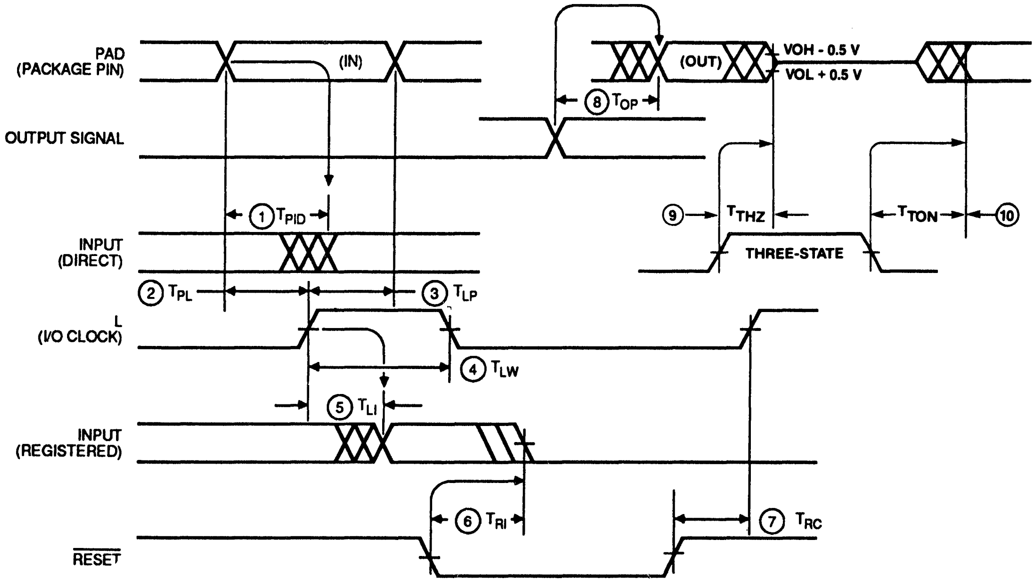
* *Refer to MIL-M-38510, Appendix C for the appropriate package drawings.*

APPLICATION GUIDELINES, SWITCHING CHARACTERISTICS – IOB

Symbol	Description	-20		-33		-50		-70 *		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
t_{PID}①	Pad (package pin) to input (direct)		20		12		8		6	ns
t_{LI}⑤	I/O Clock to input (storage)		30		20		15		11	ns
t_{PL}②	I/O Clock to pad-input setup	20		12		8		6		ns
t_{LP}③	I/O Clock to pad-input hold	0		0		0		0		ns
t_{LW}④	I/O Clock pulse width	20		12		9		7		µs
	I/O Clock frequency	20		33		50		70		MHz
t_{OP}⑧	Output to pad (output enabled)		25		15		12		9	ns
t_{THZ}⑨	Three-state to pad begin hi-Z		35		25		20		15	ns
t_{TON}⑩	Three-state to pad end hi-Z		40		25		20		15	ns
t_{RI}⑥	$\overline{\text{RESET}}$ to input (storage)		50		40		30		25	ns
t_{RC}⑦	$\overline{\text{RESET}}$ to input clock		35		35		25		20	ns

Note: Timing is measured at 0.5 V_{CC} levels with 50 pF output load.

* 70 MHz specifications are preliminary.



APPLICATION GUIDELINES, PROGRAMMING CHARACTERISTICS* – MASTER MODE[1,2]

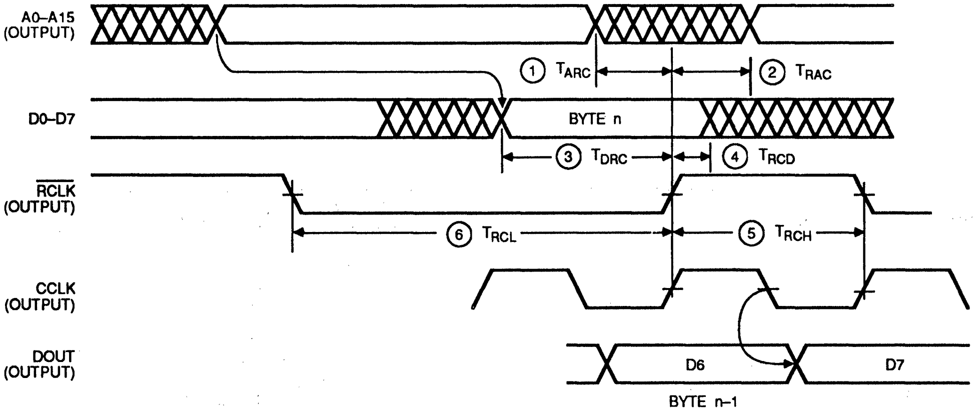
Symbol	Description		-20		-33, -50, -70*		Unit
			Min	Max	Min	Max	
t_{ARC}①	$\overline{\text{RCLK}}$	From address invalid		0		0	ns
t_{RAC}②		To address valid		300		200	ns
t_{DRC}③		To data setup	100		60		ns
t_{RCD}④		To data hold	0		0		ns
t_{RCH}⑤		$\overline{\text{RCLK}}$ HIGH	600		600		ns
t_{RCL}⑥		$\overline{\text{RCLK}}$ LOW	4.0		4.0		µs

Notes: 1. *CCLK and DOUT timing are the same as for slave mode.*

2. *At power up, V_{CC} must rise from 3.5 to V_{CC} minimum in less than 10 ms.*

* *All programming characteristics are not directly tested.*

* 70 MHz specifications are preliminary.



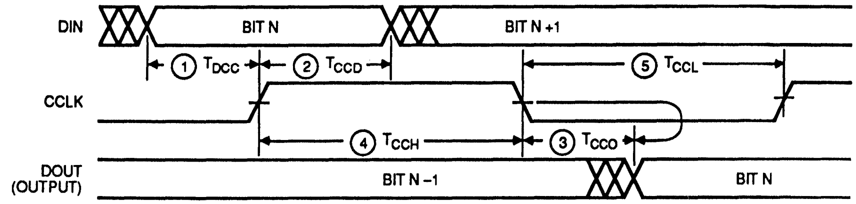
APPLICATION GUIDELINES, PROGRAMMING CHARACTERISTICS* – SLAVE MODE

Symbol	Description	-20		-33		-50, -70 *		Unit
		Min	Max	Min	Max	Min	Max	
t_{CCO}③	CCLK to DOUT		100		100		100	ns
t_{DCC}①	CCLK DIN setup	50		25		10		ns
t_{CDD}②	CCLK DIN hold	75		40		40		ns
t_{CCH}④	CCLK HIGH time	0.50		0.50		0.50		µs
t_{CCL}⑤	CCLK LOW time	0.50	1	0.50	1	0.50	1	µs
F_{CC}	CCLK frequency		1		1		1	MHz

Note: Configuration must be delayed at least 40 ms after V_{CC} minimum.

** All programming characteristics are not directly tested.*

* 70 MHz specifications are preliminary.



APPLICATION GUIDELINES, PROGRAMMING CHARACTERISTICS* – PERIPHERAL MODE

Symbol	Description		-20		-33, -50, -70*		Unit
			Min	Max	Min	Max	
t_{CA}①	Controls[2] ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$)	Active (last active input to first inactive)	0.30	10.0	0.5	1.0	μs
t_{CI}②		Inactive (first inactive input to last active)	0.25		0.5		μs
t_{CCC}③		CCLK[3]		100		75	ns
t_{DC}④		DIN setup	50		50		ns
t_{CD}⑤		DIN hold	10		5		ns

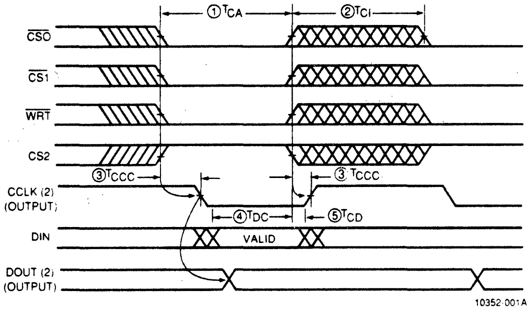
* 70 MHz specifications are preliminary.

Notes: 1. *Configuration must be delayed at least 40 ms after V_{CC} minimum.*

2. *Peripheral mode timing determined from last control signal of the logical AND of ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$) to transition to active or inactive state.*

3. *CCLK and DOUT timing are the same as for slave mode.*

* *All programming characteristics are not directly tested.*



APPLICATION GUIDELINES, SWITCHING CHARACTERISTICS – PROGRAM READBACK*

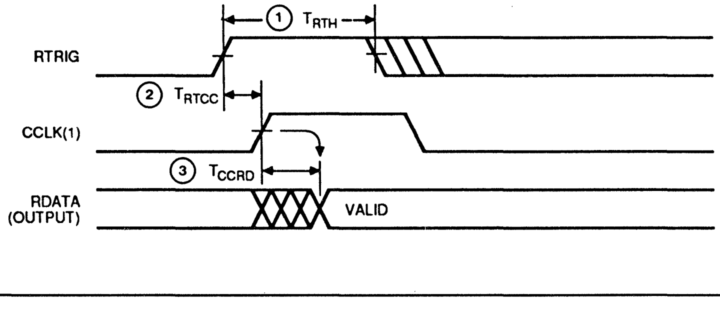
Symbol	Description		-20		-33, -50, -70*		Unit
			Min	Max	Min	Max	
t_{RTH}①	RTRIG	RTRIG HIGH	250		250		ns
t_{RTCC}②	CCLK	RTRIG setup	100		100		ns
t_{CCRD}③		RDATA delay		100		100	ns

* 70 MHz specifications are preliminary.

Notes: 1. *CCLK and DOUT timing are the same as for slave mode.*

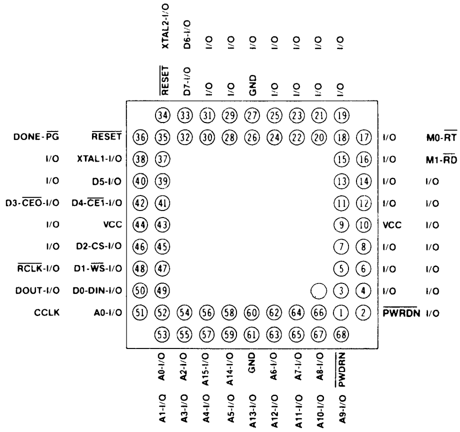
2. *DONE/PROG output/input must be HIGH (device programmed) prior to a positive transition of RTRIG (M0).*

* *All program readback characteristics are not directly tested.*

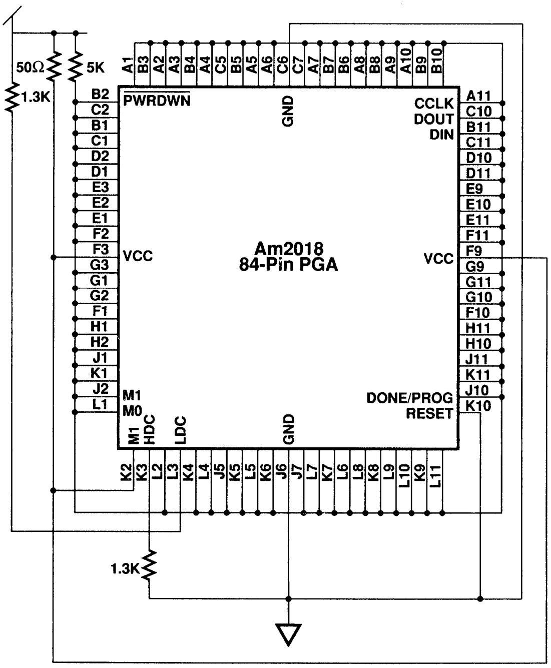


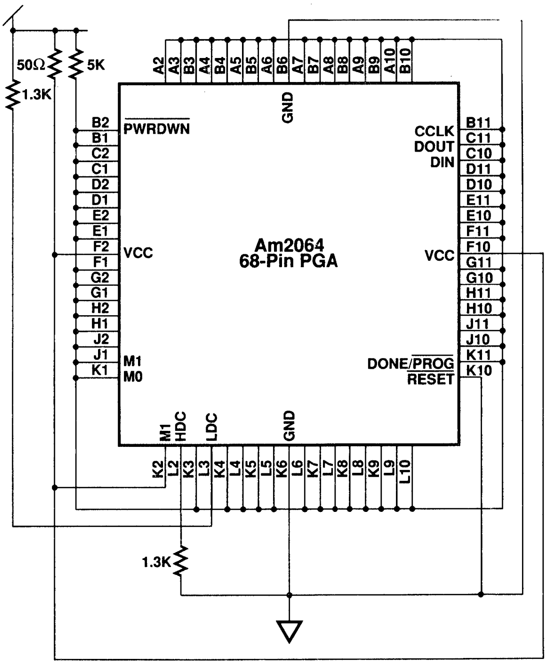
DEVICE PINOUTS*

Am2064
Logic Cell Array
68-Pin Grid Array
Top View



BURN-IN CIRCUITRY*

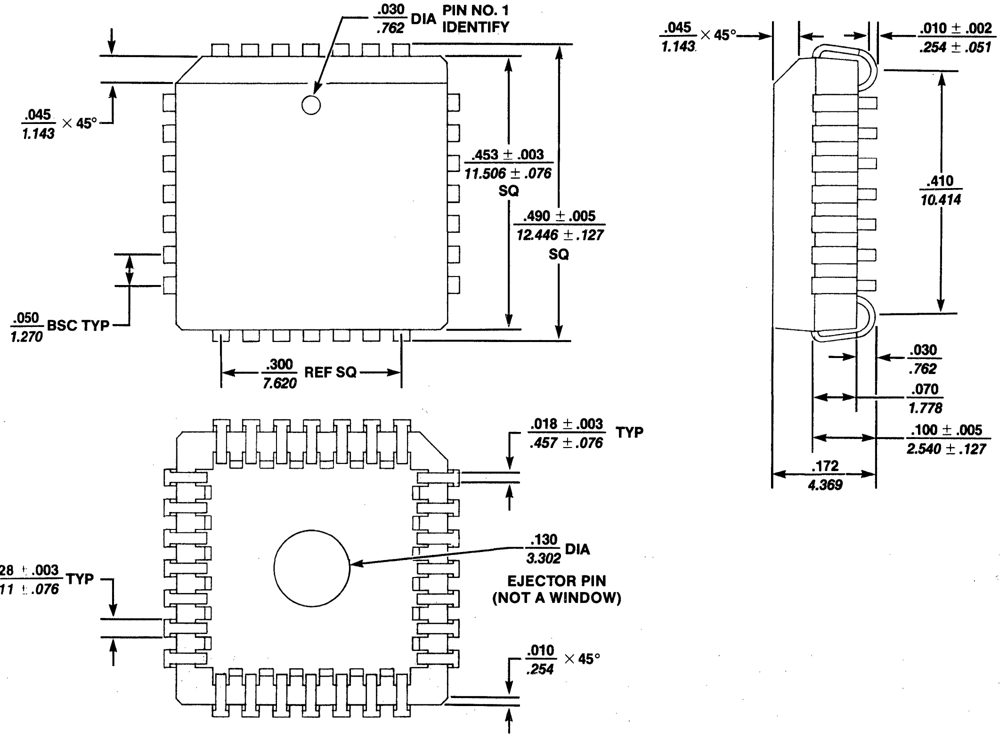

50Ω
5K
1.3K
A1 B3 A2 A3 B4 A4 C5 B5 A5 A6 C6 C7 A7 B7 B6 A8 B8 A9 A10 B9 B10
B2 C2 B1 C1 D2 D1 E3 E2 E1 F2 F3 G3 G1 G2 F1 H1 H2 J1 K1 J2 L1
PWRDWN
GND
CCLK
DOUT
DIN
A11 C10 B11 C11 D10 D11 E9 E10 E11 F11 F9 G9 G11 G10 F10 H11 H10 J11 K11 J10 K10
VCC
Am2018
84-Pin PGA
VCC
M1
M0
DONE/PROG
RESET
M1
HDC
LDC
GND
K2 K3 L2 L3 K4 L4 J5 K5 L5 K6 J6 J7 L7 K7 L6 L8 K8 L9 L10 K9 L11
1.3K


50Ω
5K
1.3K
A2 A3 B3 A4 B4 A5 B5 A6 B6 A7 B7 A8 B8 A9 B9 A10 B10
B2 B1 C2 C1 D2 D1 E2 E1 F2 F1 G2 G1 H2 H1 J2 J1 K1
PWRDWN
GND
CCLK
DOUT
DIN
B11 C11 C10 D11 D10 E11 E10 F11 F10 G11 G10 H11 H10 J11 J10 K11 K10
VCC
Am2064
68-Pin PGA
VCC
M1
M0
DONE/PROG
RESET
M1
HDC
LDC
GND
K2 L2 K3 L3 K4 L4 K5 L5 K6 L6 K7 L7 K8 L8 K9 L9 L10
1.3K

* *Unless otherwise specified all resistors are metal film and are rated for 1/8 Watt at 150°C with a build tolerance of 1% and a 5% tolerance over life.*

PHYSICAL DIMENSIONS

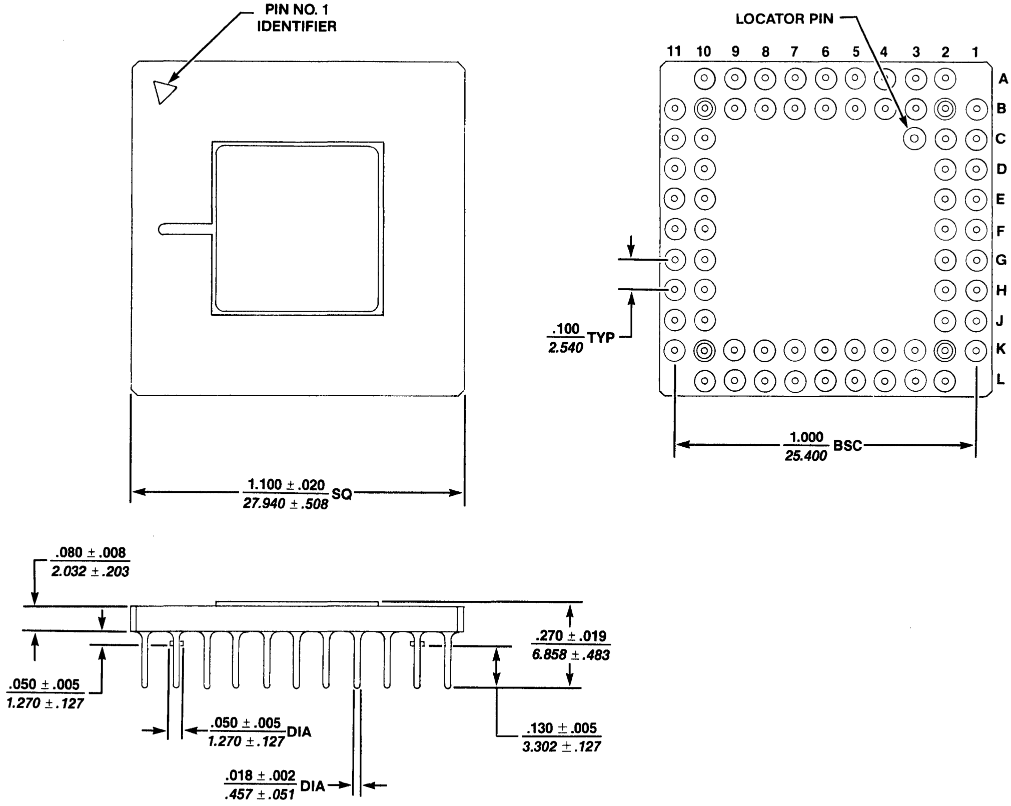
28-Pin Plastic Leaded Chip Carrier
(.451"x.451")



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PHYSICAL DIMENSIONS

68P Ceramic Pin Grid Array



10765A

UNLESS OTHERWISE SPECIFIED:

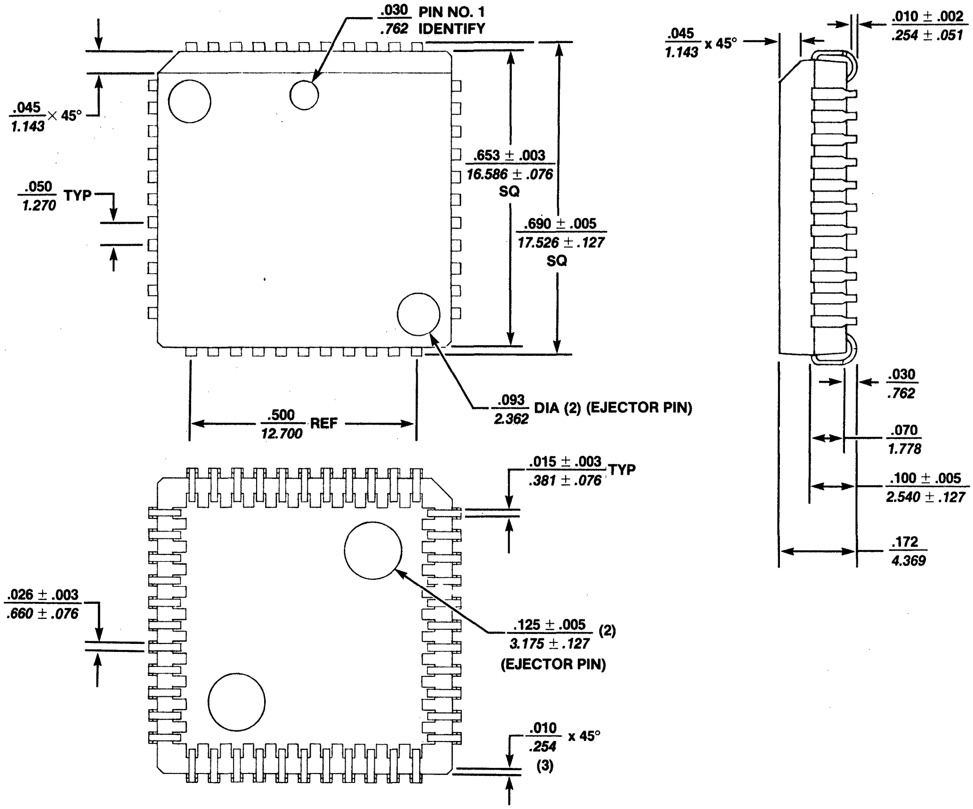
ALL DIMENSIONS MIN.-MAX. IN INCHES

ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS

ALL TOLERANCES ARE ± .007 INCHES

PHYSICAL DIMENSIONS

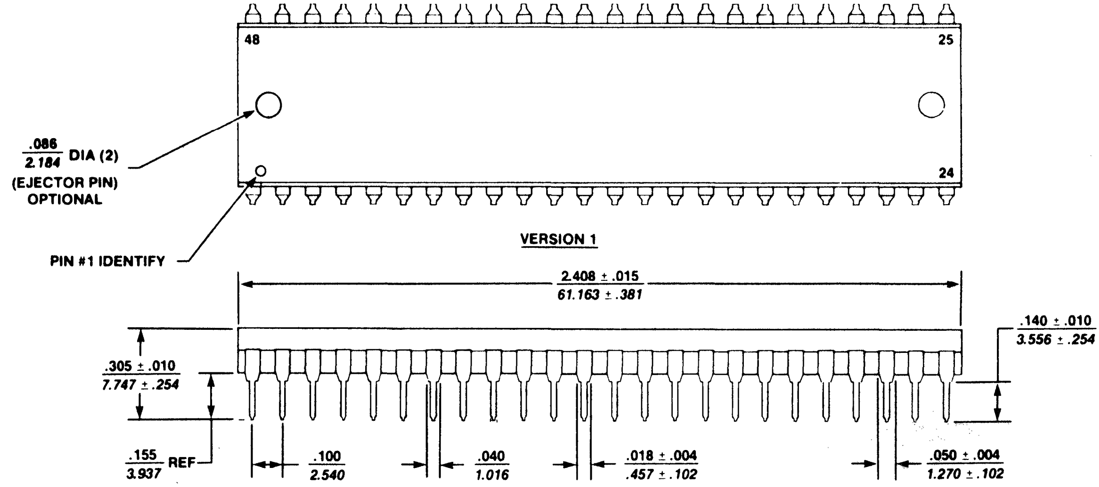
44-Pin Plastic Leaded Chip Carrier
(.650"x.650")

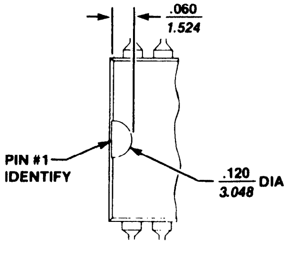


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

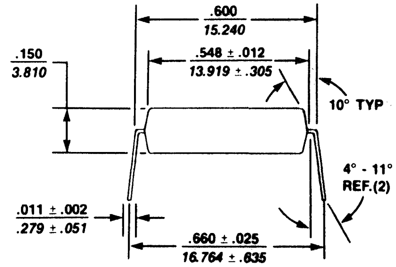
PHYSICAL DIMENSIONS

48P Molded DIP
(9/16"x2 13/32")





VERSION 2



Notes:

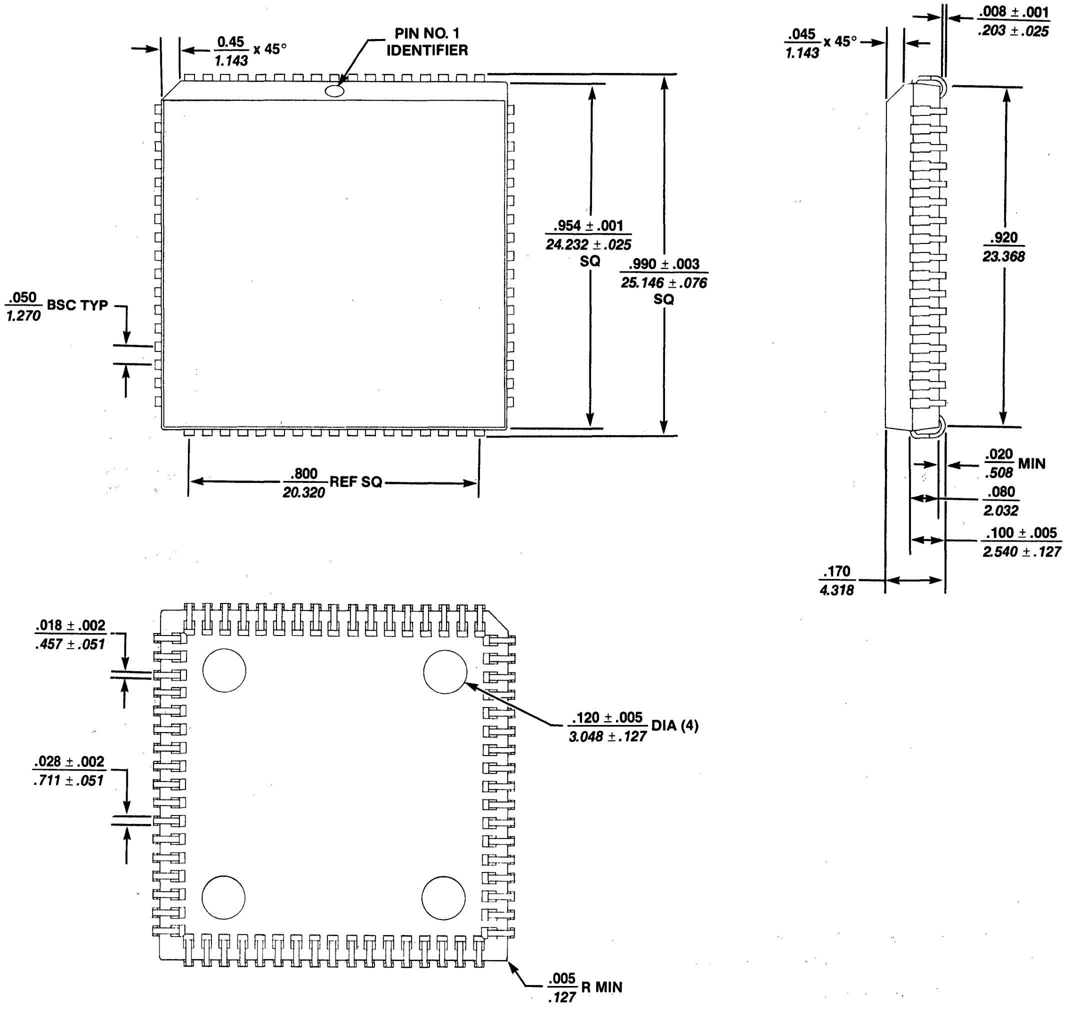
1. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2-10 mils thickness to all lead tip dimensions.
2. Both version 1 and version 2 configurations are manufactured interchangeably.
3. Ejector pin marks on version 1 are optional.

10749A

UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PHYSICAL DIMENSIONS

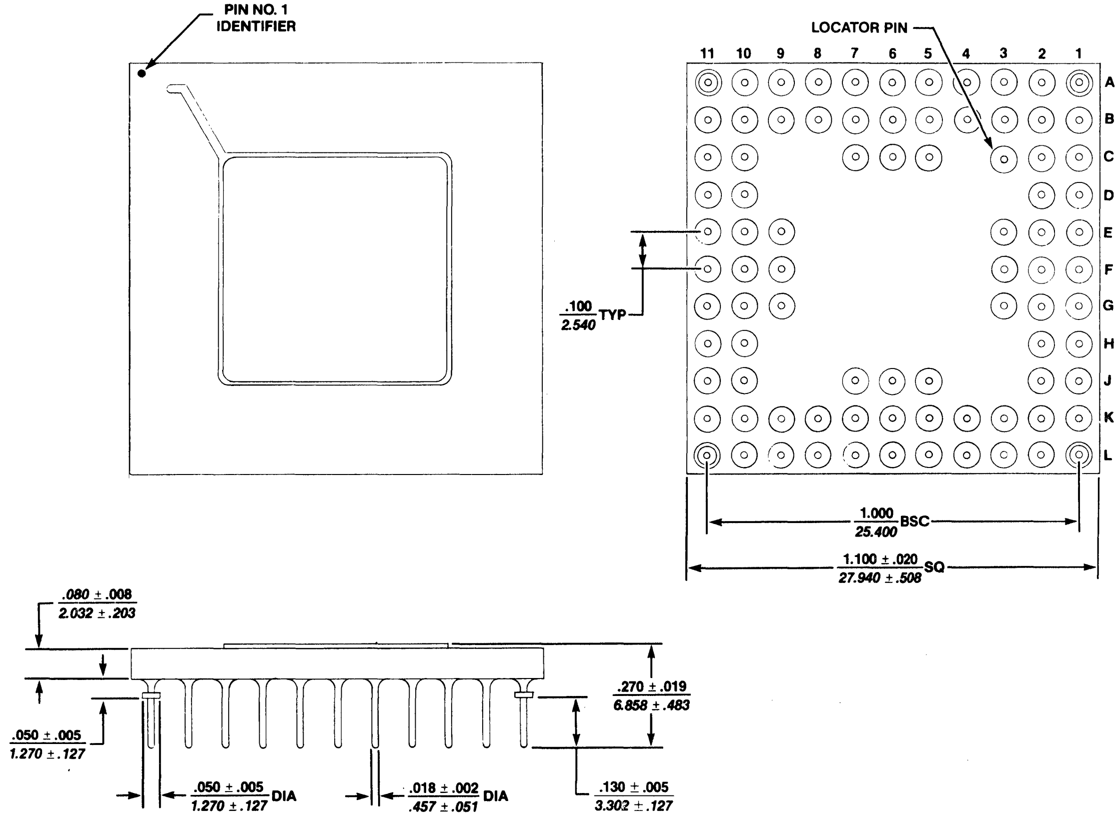
68J Plastic Leaded Chip Carrier
(.950"x.950")



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PHYSICAL DIMENSIONS

84P Ceramic Pin Grid Array

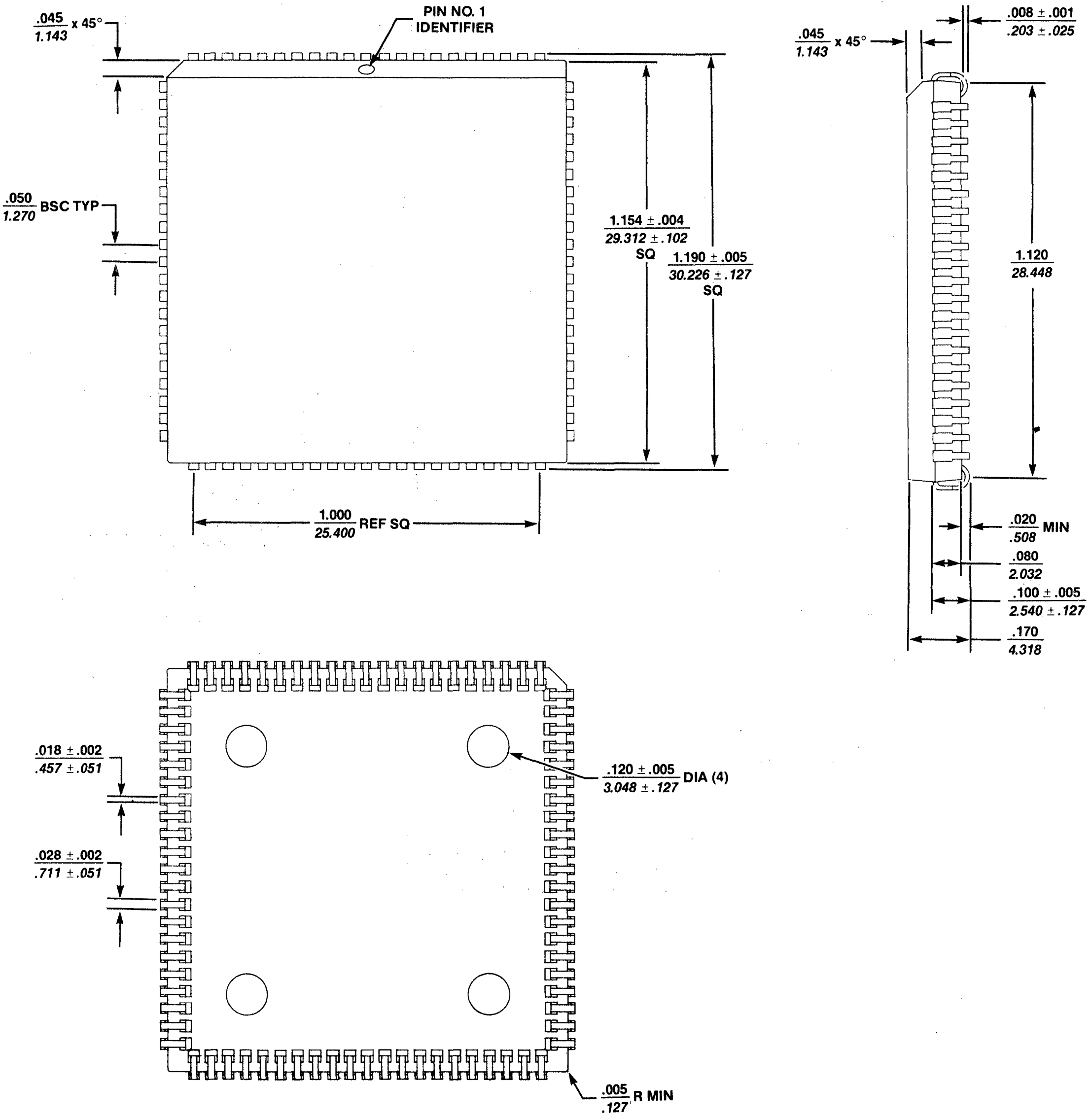


10766A

UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PHYSICAL DIMENSIONS

84J Plastic Leaded Chip Carrier
(1.154"x1.154")



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Preliminary

Am3020/3030/3042/3064/3090

Am3000 Series Family of Programmable Gate Arrays

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Second generation user-programmable gate array**
- **Flexible array architecture**
- **High performance**
  - 50, 70, 100 MHz commercial products
  - 50, 70 MHz military products
- **Improved interconnection resources**
- **Density of up to 9000 gates**
- **100% factory pre-tested**
- **Selectable configuration modes**
- **100% compatibility with AMD PGA development tools**
- **Standard PROM file interface**
- **Off the shelf availability**

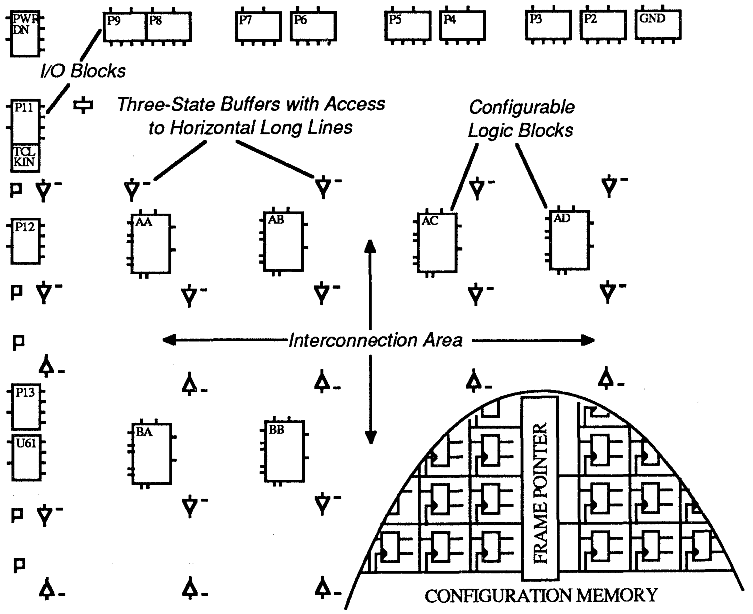
GENERAL DESCRIPTION

The Am3000 Series Logic Cell^{TM} Array (LCA) is a high-performance, second generation user-programmable gate array. The array contains three types of configurable elements that are customized in accordance with the user-defined system design; a perimeter of Input/ Output Blocks (IOBs), a core array of Configurable Logic Blocks (CLBs), and interconnection resources.

The final configuration of the three main programmable elements is determined by the user and easily implemented by AMD user-programmable gate array design tools.

AMD's development tools let users produce a complete design, from schematic capture through device customization, on an IBM PC-AT^{TM} compatible computer. LCA macro libraries and interface software are also available to support schematic capture and simulation on popular CAE workstations.

BLOCK DIAGRAM



Logic Cell is a trademark of Xilinx, Inc.
IBM PC-AT is a trademark of International Business Machines Corp.

Publication # 10642 Rev. B Amendment /0
Issue Date: August 1989

PRODUCT SELECTOR GUIDE

BASIC ARRAY	LOGIC CAPACITY (USABLE GATES)	CONFIGURABLE LOGIC BLOCKS	USER I/OS	PROGRAM DATA (BITS)
Am3020	2000	64	64	14779
Am3030	3000	100	80	22176
Am3042	4200	144	96	30784
Am3064	6400	224	120	46064
Am3090	9000	320	144	64160

ORDERING INFORMATION
Standard Products

AMD standard products are available in several packages and operating ranges. The ordering number (Valid Combination) is formed by a combination of:

a. Device Number
b. Speed Option (if applicable)
c. Package Type
d. Temperature Range
e. Number of Pins

AM3020 -50 J C 068

e. NUMBER OF PINS
068 (68 pins)
084 (84 pins)
132 (132 pins)*
175 (175 pins)

d. TEMPERATURE RANGE
C = Commercial (0°C to 70°C)
I = Industrial (-40°C to +85°C)

c. PACKAGE TYPE
J = Plastic Leaded Chip Carrier
G = Pin Grid Array

b. SPEED OPTION
-50 (50 MHz toggle rate)
-70 (70 MHz toggle rate)
-100 (100 MHz toggle rate)

a. DEVICE NUMBER/DESCRIPTION
Am3000 Series of Programmable Gate Arrays
Am3020 (2000 gates)
Am3030 (3000 gates)
Am3042 (4200 gates)
Am3064 (6400 gates)
Am3090 (9000 gates)

Package Availability Guide

	68-pin PLCC	68-pin PGA	84-pin PLCC	84-pin PGA	175-pin PGA
3020	x	x	x	x	
3030			x	x	
3042			x	x	
3064	**	**	**	**	**
3090					x

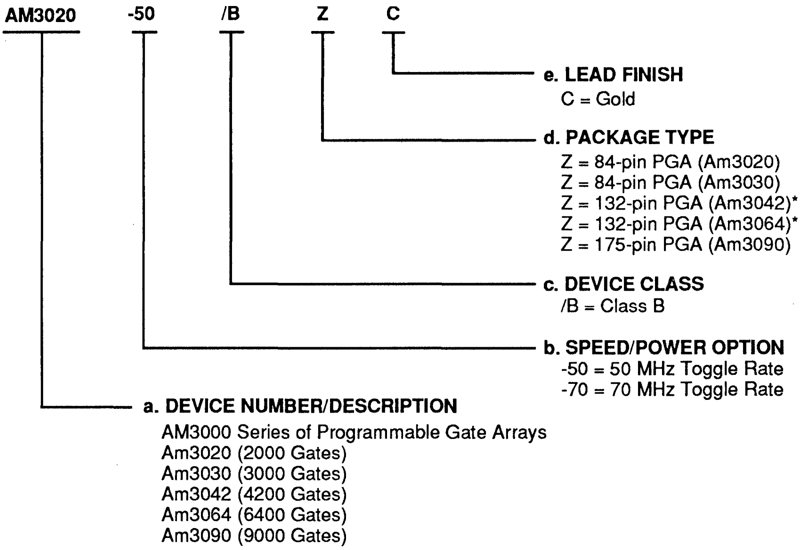
** Package types for the Am3064 are to be determined

*In development

ORDERING INFORMATION
Military APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) for APL products is formed by a combination of:

a. Device Number
b. Speed/Power Option (if applicable)
c. Device Class
d. Package Type
e. Lead Finish



e. LEAD FINISH
C = Gold

d. PACKAGE TYPE
Z = 84-pin PGA (Am3020)
Z = 84-pin PGA (Am3030)
Z = 132-pin PGA (Am3042)*
Z = 132-pin PGA (Am3064)*
Z = 175-pin PGA (Am3090)

c. DEVICE CLASS
/B = Class B

b. SPEED/POWER OPTION
-50 = 50 MHz Toggle Rate
-70 = 70 MHz Toggle Rate

a. DEVICE NUMBER/DESCRIPTION
AM3000 Series of Programmable Gate Arrays
Am3020 (2000 Gates)
Am3030 (3000 Gates)
Am3042 (4200 Gates)
Am3064 (6400 Gates)
Am3090 (9000 Gates)

Valid Combinations	
Am3020-50	/BZC
Am3020-70	
Am3030-50	
Am3030-70	
Am3042-50	
Am3042-70	
Am3064-50	
Am3064-70	
Am3090-50	
Am3090-70	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A Tests consist of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

Military Burn-in

Military burn-in is in accordance with the current revision of MIL–STD–883, Test Method 1015, Conditions A through E. Test conditions are selected at AMD's option.

*In development

3

ARCHITECTURE

Functional Description

The perimeter of configurable IOBs provides a programmable interface between the internal logic array and the device package pins. The array of CLBs performs user-specified logic functions. The interconnections are programmed to form networks, carrying logic signals among blocks. This is analogous to printed circuit board traces connecting MSI/SSI packages.

The logic functions of these blocks are determined by programmed look-up tables. Functional options are performed by program-controlled multiplexers. Interconnecting networks between blocks are composed of metal segments joined by program-controlled pass transistors. These LCA functions are activated by a configuration bit stream that is loaded into an internal, distributed array of configuration memory cells. The configuration bit stream is loaded into the LCA device at power-up and can be reloaded on command. The LCA device includes logic and control signals for automatic or passive configuration. Configuration data can be either bit serial or byte parallel. The PGA Development System generates the configuration bit stream used to configure the LCA device. The memory loading process is independent of the user logic functions.

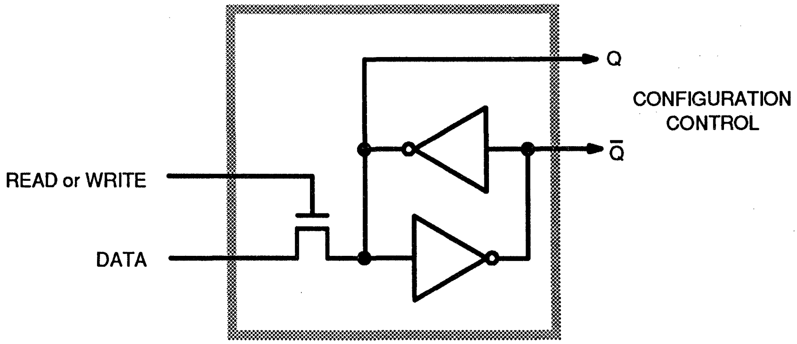
Configuration Memory

The static memory cell used for the LCA's configuration memory has been designed specifically for high reliability and noise immunity, ensuring integrity even under adverse conditions. Static memory provides the best combination of high density, high performance, high reliability, and comprehensive testability. As shown below, the basic memory cell consists of two CMOS inverters and a pass transistor, which is used for writing and reading cell data. The cell is only written during configuration and only read during readback. During normal operation, the pass transistor is off and does not affect the stability of the cell. This is quite different from the operation of conventional memory devices, in which the cells are frequently read and re-written.

A static configuration memory cell is loaded with one bit of the configuration bit stream and controls one data selection in the LCA device. The memory cell outputs Q and $\overline{Q}$ use full Ground and V_{CC} levels and provide continuous, direct control. The additional capacitive load, together with the absence of address decoding and sense amplifiers, gives the cell high stability.

Due to the structure of the configuration memory cells, they are not affected by extreme power supply excursions or very high levels of alpha particle radiation. No soft errors have been observed in reliability testing, even in the presence of very high doses of alpha radiation.

The method of loading the configuration data is selectable. Two methods use serial data, while three use byte-wide data. The internal configuration logic uses framing information, which is embedded in the configuration data by the PGA Development System, to direct memory cell loading. The serial data framing and length count preamble provide synchronous, serial, or daisy-chained compatibility with various AMD programmable gate arrays.



10642-019A

Figure 1. Basic Memory Cell

Input/Output Blocks

Each user-configurable IOB, shown below, provides an interface between the device's external package pin and the internal user logic. Each IOB includes both registered and direct input paths and each provides a programmable three-state output buffer that can be driven by a registered or direct output signal. Configuration options allow a choice of polarity on the output and three-state control signals, a controlled slew rate, and a high impedance pull-up. Each input circuit provides input clamping diodes for electrostatic protection, and circuits to inhibit latch-up produced by input currents.

The input buffer portion of each IOB provides threshold detection to translate external signals applied to the package pin to internal logic levels. The global input-buffer threshold of the IOBs can be programmed for TTL or CMOS voltage levels. The buffered input signal drives the data input of a storage element that can be configured as a positive edge-triggered D flip-flop, or a level-transparent latch. The sense of the clock can be inverted (negative edge/high transparent) as long as all IOBs on the same clock net use the same clock sense. Clock/load signals, IOB pins .ik and .ok, can be chosen from either of two available metal lines along each die edge. I/O storage elements are reset during configuration or by the active low chip $\overline{\text{RESET}}$ input. Both direct input from IOB pin .i, and registered input from IOB pin .q signals are available for interconnect.

For reliable operation, inputs should have transition times less than 100 ns and should not be left undriven, or floating. Unused CMOS input-pin circuits can be at threshold and produce oscillations. This produces additional power dissipation and system noise. A typical hysteresis of about 300 mV reduces input noise sensitivity. Each user IOB includes a programmable high impedance pull-up resistor that can be selected by the bit stream and which provides a constant HIGH for otherwise undriven package pins. Although the LCA device provides circuitry for input protection against electrostatic discharge, normal CMOS handling precautions should be observed.

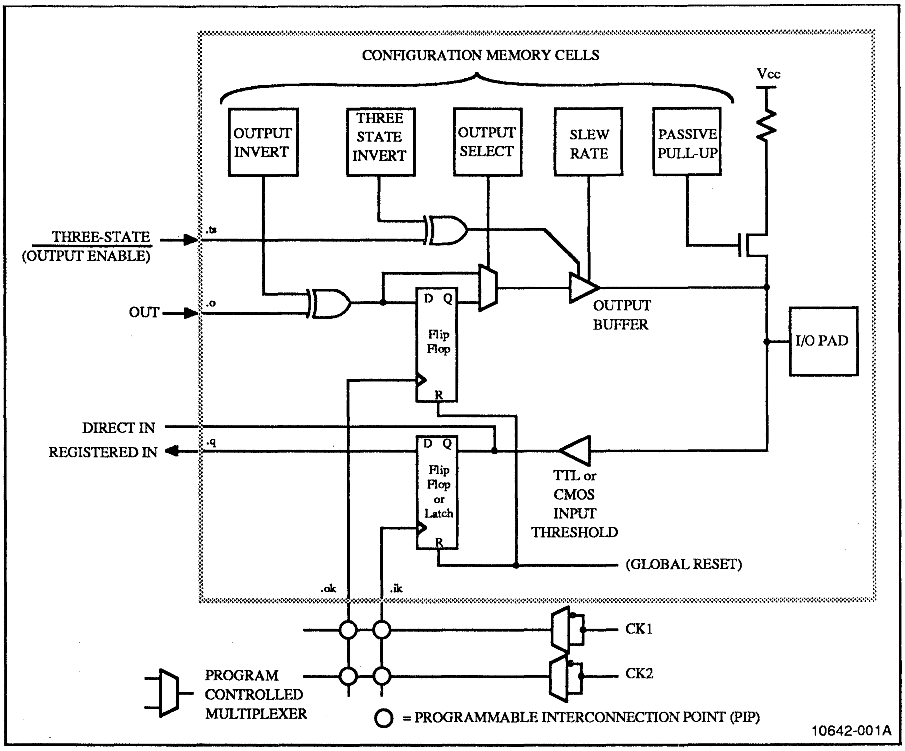


Figure 2. Input/Output Block

Flip-flop loop delays for the IOB and logic block flip-flops are about 3 ns. This increases reliability, especially for asynchronous clock and data conditions. Short loop delays minimize the probability of a metastable condition, which can result from assertion of the clock during data transitions. Because of the short loop delay in LCA devices, the I/O flip-flops can be used to synchronize external signals applied to the device. Once synchronized in the IOB, the signals can be used internally without regard to their clock-relative timing, except as it applies to the internal logic and routing path delays.

Output buffers of the IOBs provide CMOS-compatible 4 mA source-or-sink drive for high fan-out CMOS or TTL compatible signal levels. The network driving IOB pin .o becomes the registered or direct data source for the output buffer. The three-state control signal, IOB pin .ts, can control output activity. An open-drain type output can be obtained by using the same signal for driving the output and three-state signal nets, so that the buffer output is enabled only for a LOW.

The configuration memory cells, shown in Figure 2, control the optional output register and logical signal inversion, as well as the three-state and slew rate configuration bits. A choice of two clocks is available on each die edge. All user inputs are programmed for TTL or CMOS thresholds.

The IOB includes input and output storage elements and the following I/O options selected by configuration memory cells.

- Logical **inversion of the output** is controlled by one configuration bit per IOB.
- Logical **three-state control** of each IOB output buffer is determined by the states of the configuration data bits that turn the buffer on/off or select the output buffer three-state control interconnection, IOB pin .ts. When this IOB output control signal is HIGH, or logic 1, the buffer is disabled and the package pin is high impedance. Inversion of the buffer three-state control logic sense, output enable, is controlled by an additional configuration data bit.
- **Direct or registered output** is selectable for each IOB. The register uses a positive-edge, clocked flip-flop. The clock source, IOB pin .ok, can be supplied by either of two metal lines, which are available along each die edge. Each of these lines is driven by an invertible buffer.
- Increased **output transition speed** can be selected to satisfy critical nets. Slower transitions reduce capacitive load peak currents of non-critical outputs and minimize system noise.
- A high impedance **pull-up resistor** can be used to prevent floating, unused inputs.

The table below summarizes the I/O options.

INPUTS	OUTPUTS
Direct	Direct/registered
Flip-flop/latch	Inverted/true
CMOS/TTL threshold (chip inputs)	Full speed/slew limited
Optional pull-up resistor	Optional three-state control

Configurable Logic Blocks

CLBs are the functional elements from which the user's logic is constructed. The logic blocks are arranged in a matrix within the perimeter of IOBs. The Am3020 has 64 such blocks arranged in eight rows and eight columns. The PGA Development System compiles the configuration data, which defines the operation and interconnection of each block. Users can define CLBs and their interconnecting networks by automatic translation from a schematic capture logic diagram or, optionally, by installing library or user macros.

Each CLB has a combinational logic section, two flip-flops, and an internal control section. As shown in the following figure, there are five logic inputs (.a, .b, .c, .d, and .e), a common clock input (.k), an asynchronous direct reset input (.rd), and a clock enable (.ec). All can be driven from the interconnection resources adjacent to the blocks. Each CLB also has two outputs (.x and .y) that can drive interconnection networks.

Data input for either flip-flop within a CLB is supplied from the F or G function outputs of the combinational logic, or the direct data input, .di. Both flip-flops in each CLB share the asynchronous reset, .rd, which, when enabled and HIGH, is dominant over clocked inputs. All flip-flops are reset by the active-LOW chip input, $\overline{\text{RESET}}$, or during the configuration process. The flip-flops share the clock enable (.ec), which, when LOW, recirculates the present states of the flip-flops and inhibits response to the data-in or combinational function inputs on a CLB. The user can enable these control inputs and select their sources. The user also can select the clock net input (.k) and its active sense in each logic block. This programmable inversion eliminates the need to route both phases of a clock signal throughout the device. Flexible routing allows use of common or individual CLB clocking.

The combinational logic portion of the CLB uses a 32-by-1 look-up table to perform Boolean functions. Variables selected from the five logic inputs and two internal block flip-flops are used as table address inputs. The combinational propagation delay through the network is independent of the logic function generated and is spike free for changes in single input variables.

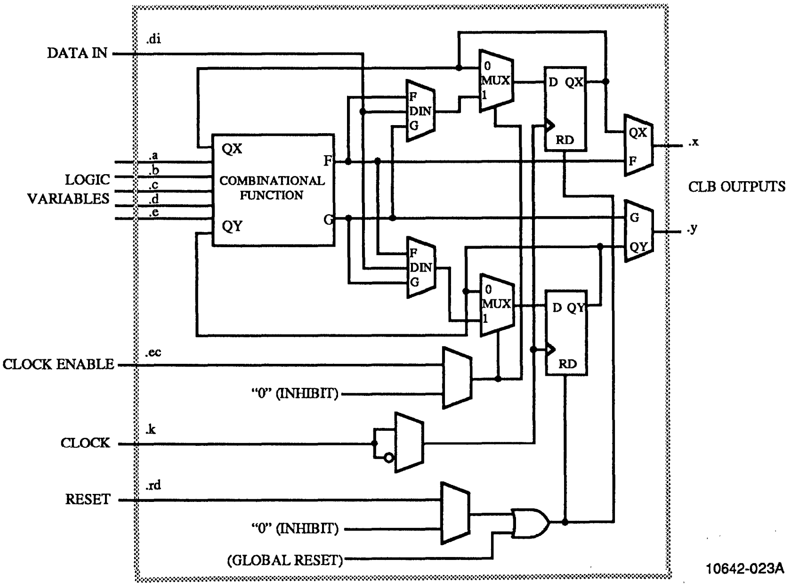


Figure 3. Configurable Logic Block

This technique can generate a single function of five variables, as shown below.

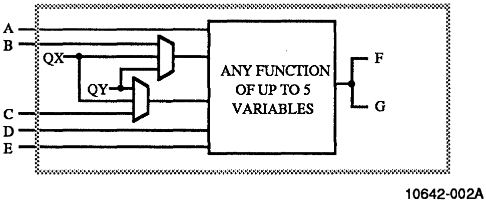


Figure 4. Combinatorial Logic Option 1

It can also generate any two logic functions of up to four variables each.

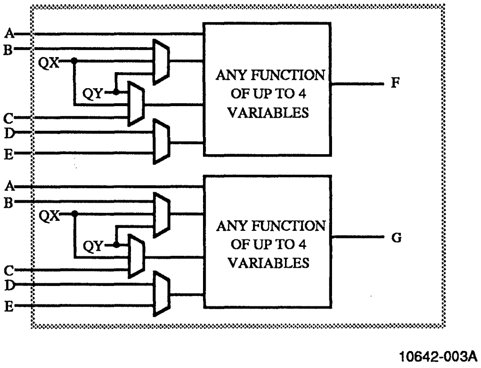
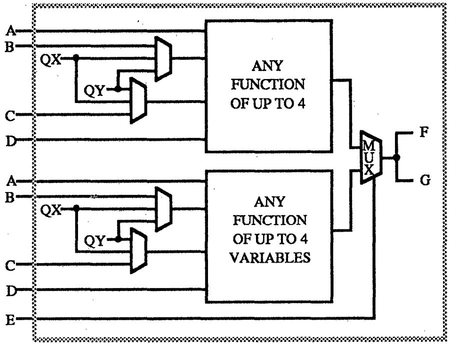

A
B
QX
QY
C
D
E
ANY FUNCTION OF UP TO 4 VARIABLES
F
A
B
QX
QY
C
D
E
ANY FUNCTION OF UP TO 4 VARIABLES
G
10642-003A

Figure 5. Combinatorial Logic Option 2

It can also generate some functions of seven variables, as shown in the next figure.



10642-004A

Figure 6. Combinatorial Logic Option 3

The partial functions of six or seven variables are generated by the input variable, .e, which dynamically selects between two functions of four different variables. For the two functions of four variables each, the independent results, F and G, can be used as data inputs to either flip-flop or either logic block output. For the single function of five variables and merged functions of six or seven variables, the F and G outputs are identical. Symmetry of the F and G functions and the flip-flops helps optimize the routing of the networks connecting the logic blocks and IOBs.

The next figure shows a modulo 8 binary counter with parallel enable. It uses one CLB of each type.

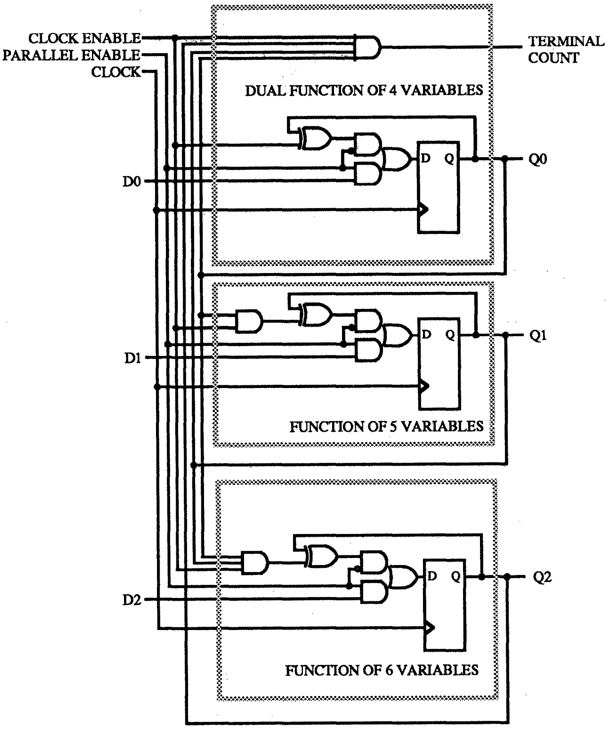


Figure 7. C8BCP Macro (modulo 8 binary counter with parallel enable and clock enable)

INTERCONNECTIONS

Programmable Interconnections

Programmable interconnection resources in the LCA device provide routing paths to connect inputs and outputs of the I/O and logic blocks into logical networks. Interconnections between blocks are composed of two-layer grid of metal segments. Specially designed pass transistors, each controlled by a configuration bit, form programmable interconnect points (PIPs) and switching matrices used to implement the necessary connections between selected metal segments and block pins. The figure below provides an example of a routed net.

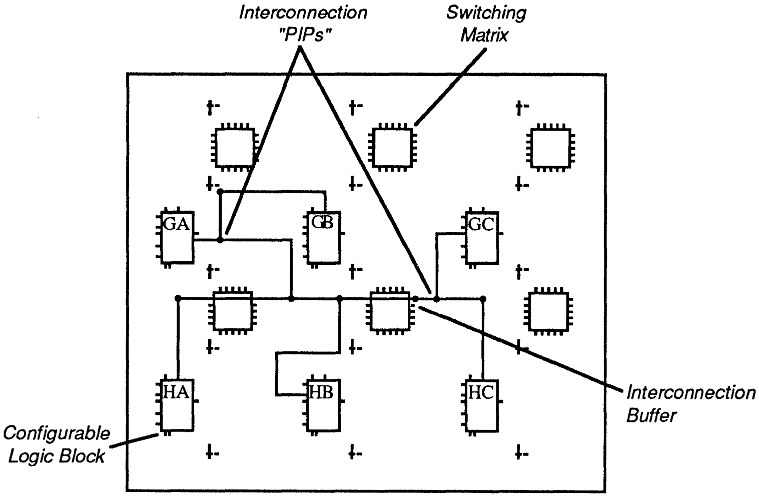


Figure 8. Routing Resources

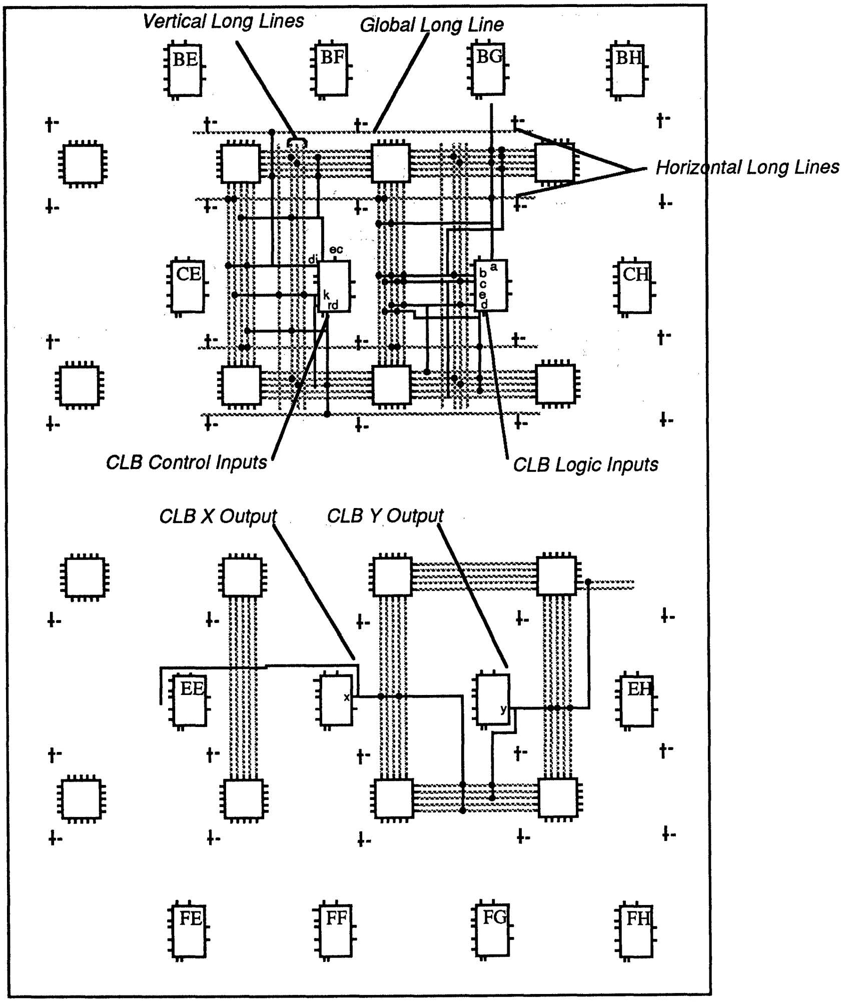
The PGA Development System automatically routes these interconnections. Interactive routing can also be done to optimize the design. The inputs of the CLB or IOB are multiplexers that can be programmed to select an input network from the adjacent interconnection segments.

Note: The switch connections to block inputs are usable only for input connection, and not for routing, because they are unidirectional (as are block outputs).

The figure below illustrates routing access to logic block input variables, control inputs, and block outputs.

Three types of metal resources are available for network interconnections.

- General-Purpose Interconnection
- Direct Connection
- Long Lines



10642-005A

Figure 9. Routing Access to Inputs, Outputs

General-Purpose Interconnections

A general-purpose interconnection, as shcwn below, consists of a grid of five horizontal and five vertical metal segments located between the rows and columns of CLBs and IOBs. These segments can be connected through switch matrices to form networks for CLB and IOB inputs and outputs.

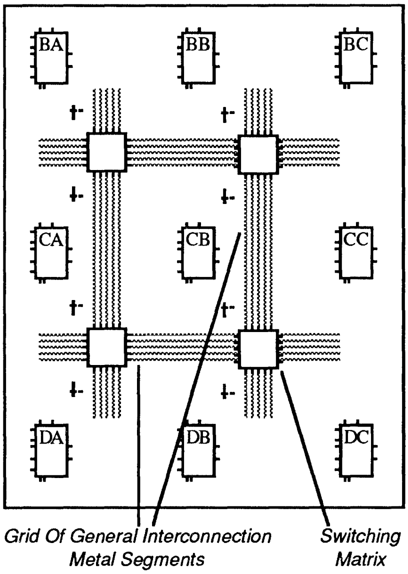


Figure 10. General Purpose Interconnections

Each segment is the height or width of a logic block. Switching matrices join the ends of these segments and allow programmed interconnections between the metal grid segments of adjoining rows and columns. The switches of an unprogrammed device are all non-conducting. The connections through the switch matrix can be made by automatic routing, or by using Editnet to select the desired pairs of matrix pins that are to be connected or disconnected. The legitimate switching matrix combinations for each pin are shown in the next figure, and may be highlighted by the use of the SHOW MATRIX command.

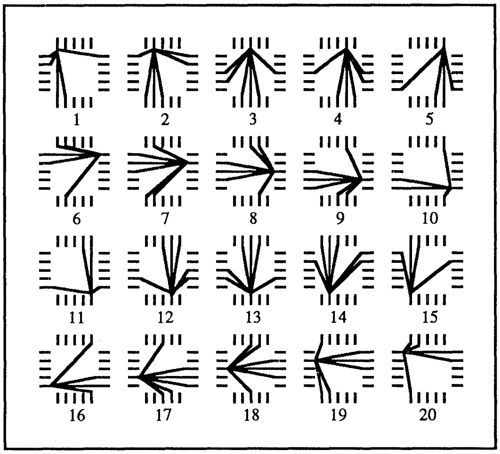


Figure 11. Switch Matrix Interconnection Options

Special buffers in the general interconnection areas are inserted automatically to provide periodic signal isolation and restoration, thus improving performance of lengthy nets. The interconnection buffers can propagate signals in either direction on a general interconnection segment. These bidirectional buffers are above and to the right of the switching matrices, and can be highlighted by the use of the SHOW MATRIX command. The other PIPs adjacent to the matrices are gateways to and from long lines.

The PGA Development System automatically defines the buffer direction based on the location of the interconnection network source. The delay calculator of the PGA Development System automatically calculates and displays the block, interconnection, and buffer delays for the selected paths. It can also generate the simulation net list with a worst-case delay model.

Direct Interconnections

A direct interconnection, shown below, provides the most efficient implementation of networks between adjacent CLBs or IOBs. The .x and .y outputs of each CLB have single contact, direct access to inputs of adjacent CLBs.

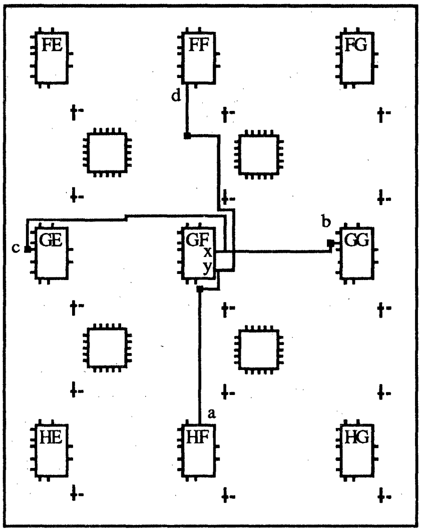


Figure 12. Direct Interconnection

Signals routed from block to block by direct interconnection show minimum interconnection propagation and use no general interconnection resources. For each CLB, the .x output can be connected directly to the .b input of the CLB to its right, and to the .c input of the CLB to its left. The .y output can use a direct interconnection to drive the .d input of the block above, and the .a input of the block below.

Direct interconnection should be used to maximize the speed of high performance portions of logic. Where CLBs are adjacent to IOBs, a direct connection is provided alternately to the IOB inputs (.i) and outputs (.o) on all four edges of the die. The right edge provides additional connections from CLB outputs to adjacent IOBs. Direct interconnections of IOBs and CLBs are shown in the next figure.

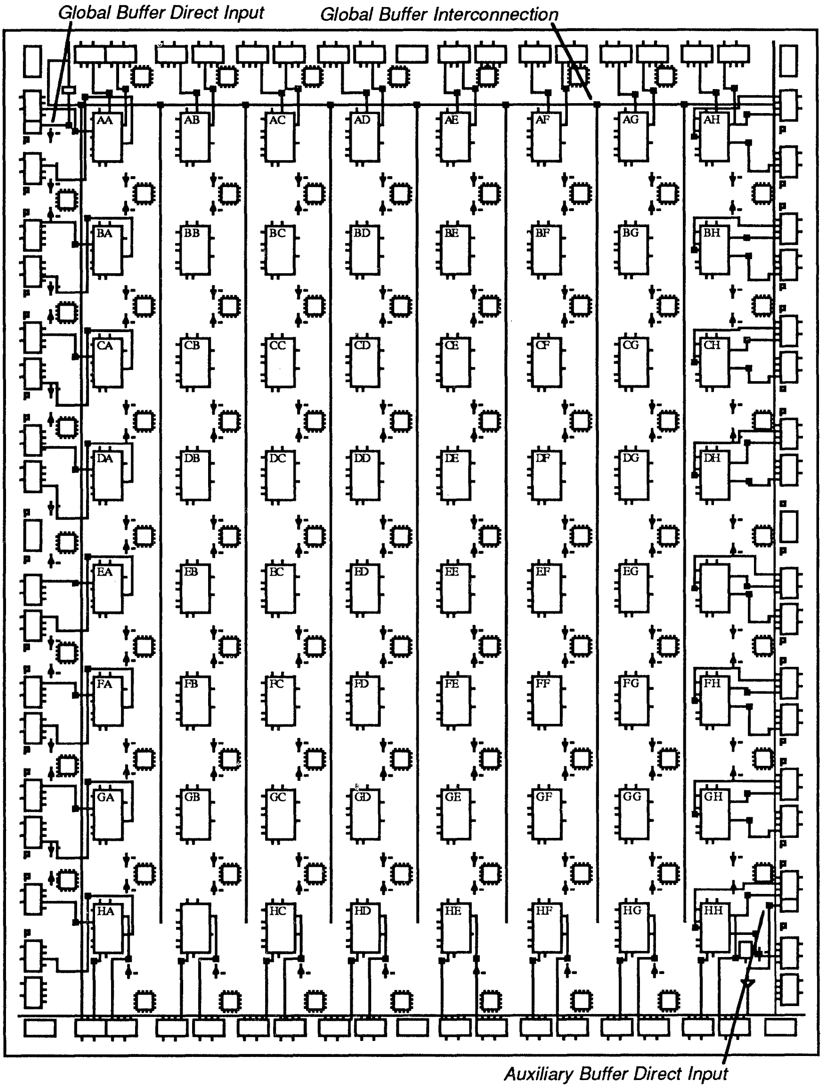


Figure 13. IOB and CLB Direct Interconnections

Long Lines

Long lines, which bypass the switch matrices, are intended primarily for signals that must travel a long distance, or that must have minimum skew among multiple destinations. Long lines, shown below, run the height or width of the interconnection area. Each interconnection column has three vertical long lines, and each interconnection row has two horizontal long lines. Two additional long lines are adjacent to the outer sets of switching matrices. On the Am3020, the outermost long lines are connectable half-length lines. In all other devices, the vertical long lines in each column are also connectable half-length.

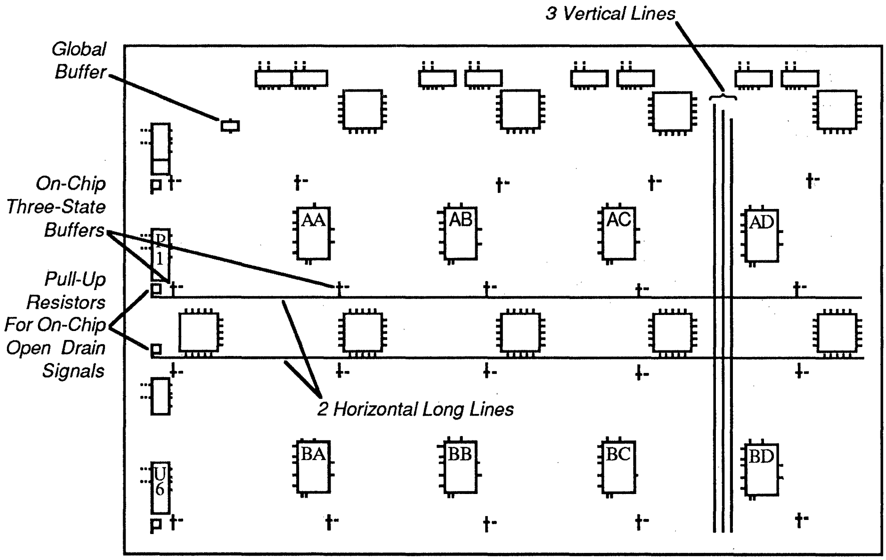

3 Vertical Lines
Global Buffer
On-Chip Three-State Buffers
Pull-Up Resistors For On-Chip Open Drain Signals
2 Horizontal Long Lines
AA
AB
AC
AD
BA
BB
BC
BD
P1
U6

Figure 14. Long Lines

Horizontal and vertical long lines provide high fan-out, low-skew signal distribution in each row and column. The programmable interconnection of long lines is provided at the edges of the routing area. Long lines can be driven by a CLB or IOB output on a column-by-column basis. This provides a common low skew control or clock line within each column of logic blocks. Interconnections of these long lines are shown in the following figure. Isolation buffers are provided at each input to a long line and are enabled automatically by the PGA Development System when a connection is made.

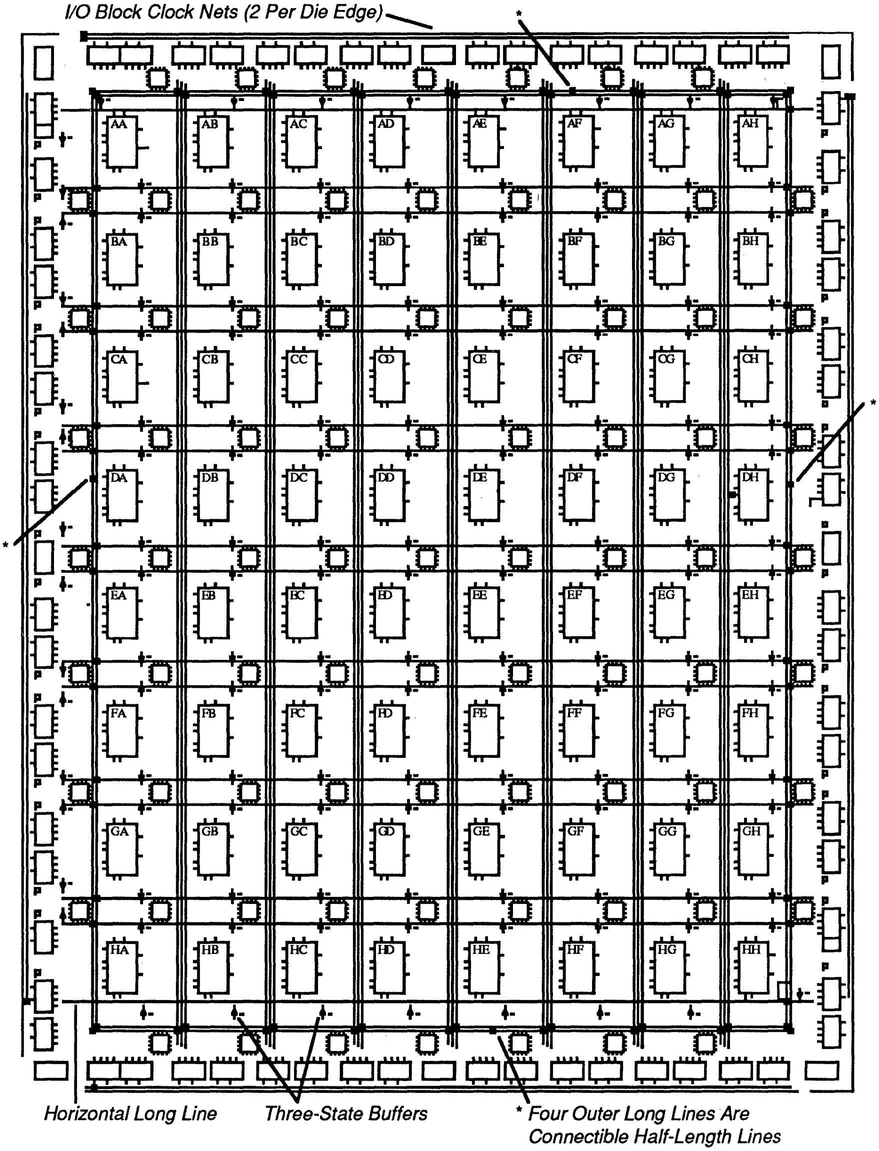


Figure 15. Interconnection of Long Lines

A buffer in the upper left corner of the LCA chip drives a global net available to all .k inputs of logic blocks. Using this global buffer for a clock signal provides a skew free, high fan-out, synchronized clock for use at any, or all, of the I/O and logic blocks. Configuration bits for the .k input to each logic block can select this global line or another routing resource as the clock source for its flip-flops. This net can also be programmed to drive the die edge clock lines for IOB use. TCLKIN is an enhanced speed, CMOS threshold, direct access to this buffer that is available at the second pad from the top of the left die edge.

A buffer in the lower right corner of the array drives a horizontal long line, which can drive programmed connections to a vertical long line in each interconnection column. This alternate buffer also is low skew and high fan-out. The network formed by this alternate buffer's long lines can be selected to drive the .k inputs of the logic blocks. The CMOS threshold, high-speed access to this buffer, BCLKIN, is at the third pad from the bottom of the right die edge.

Internal Busses

A pair of three-state buffers are located adjacent to each CLB. These let logic drive the horizontal long lines. Any three-state buffer input can be selected to drive the horizontal long line bus by applying a low logic level on its three-state control line. Logical operation of the three-state buffer controls lets them implement wide multiplexing functions. When data drives the inputs, and separate signals drive the three-state control lines, these buffers form multiplexers (three-state buses), as shown below. In this case, care must be used to prevent contention through multiple active buffers of conflicting levels on a common line.

$$Z = D_A \bullet \overline{A} + D_B \bullet \overline{B} + D_C \bullet \overline{C} + \ldots + D_N \bullet \overline{N}$$

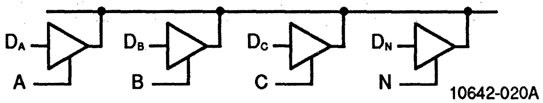
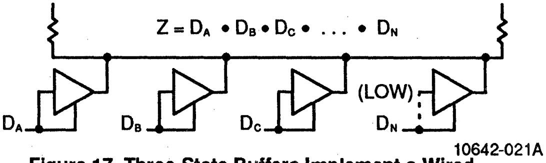


Figure 16. Three-State Buffers Implement a Multiplexer

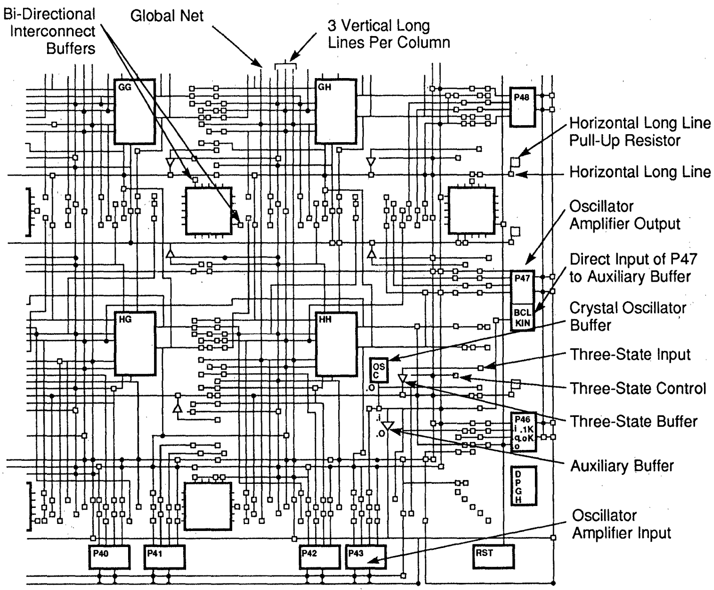
Control of the three-state input by the same signal that drives the buffer input creates an open drain wired-AND function, as shown below. A logical HIGH on both buffer inputs creates a high impedance with no contention. A logical LOW enables the buffer to drive the long line low. Pull-up resistors are available at each end of the long line to provide a HIGH output when all connected buffers are non-conducting.



$Z = D_A \bullet D_B \bullet D_C \bullet \ldots \bullet D_N$
(LOW)
D_A
D_B
D_C
D_N
10642-021A

Figure 17. Three-State Buffers Implement a Wired – AND Function

These buffers allow fast, wide gating, optimum speed, and efficient routing of high fan-out signals. The following figure shows three-state buffers, long lines, and pull-up resistors.



Bi-Directional Interconnect Buffers
Global Net
3 Vertical Long Lines Per Column
GG
GH
HG
HH
P48
Horizontal Long Line Pull-Up Resistor
Horizontal Long Line
Oscillator Amplifier Output
Direct Input of P47 to Auxiliary Buffer
P47
BCL KIN
Crystal Oscillator Buffer
OS C
.O
Three-State Input
Three-State Control
Three-State Buffer
P46 I .1K q.oK
Auxiliary Buffer
D P G H
Oscillator Amplifier Input
P40
P41
P42
P43
RST
10642–018A

Figure 18. Possible Interconnections in the Lower Right Corner of the Am3020

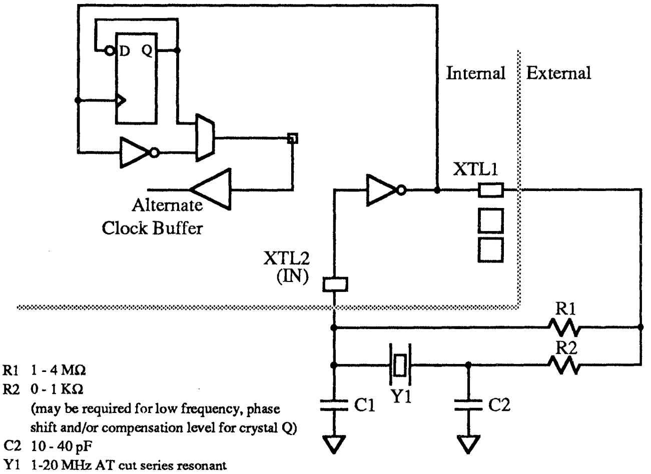
Crystal Oscillator

The previous figure also shows the location of an internal high-speed inverting amplifier that can be used as an on-chip crystal oscillator. It is associated with the auxiliary buffer in the lower right corner of the die. When the oscillator is configured as a signal source, two special user IOBs are also configured to connect the oscillator amplifier with external crystal oscillator components, as shown below.

When activated by selecting an output network for its buffer, the crystal oscillator inverter uses two of the package pins and external components to make an oscillator. An optional divide-by-two mode is available to ensure symmetry.

The oscillator circuit becomes active before configuration is complete so the oscillator can stabilize. Actual internal connection is delayed until completion of configuration. In the preceding figure the feedback resistor, R1, between output and input biases the amplifier at threshold. The value should be as large as practical to minimize loading the crystal. The inversion of the amplifier, together with the R–C networks and an AT cut series resonant crystal, produce the 360 degree phase shift of the Pierce oscillator. A series resistor, R2, can be included to increase the amplifier output impedance. This may be needed for phase shift control or crystal resistance matching, or to limit the amplifier input swing to control clipping at large amplitudes.

Excess feedback voltage can be corrected by the ratio of C2/C1. The amplifier can be used from 1 MHz to one-half the specified CLB toggle frequency. Used at frequencies below 1 MHz, the amplifier may require individual characterization with respect to a series resistance. Crystal oscillators operating at frequencies above 20 MHz usually require a crystal that operates in a third overtone mode, in which the fundamental frequency must be suppressed by the R–C networks. When the oscillator inverter is not used, these IOBs and their package pins are available for general user I/O.



R1 1 - 4 MΩ
R2 0 - 1 KΩ
(may be required for low frequency, phase shift and/or compensation level for crystal Q)
C1, C2 10 - 40 pF
Y1 1-20 MHz AT cut series resonant

	68 Pin	84 Pin		132 Pin	175 Pin
	PLCC	PLCC	PGA	PGA	PGA
XTL1 (Out)	47	57	J11	P13	T14
XTL2 (In)	43	53	L11	M13	P15

10642-008A

Figure 19. Crystal Oscillator

PROGRAMMING

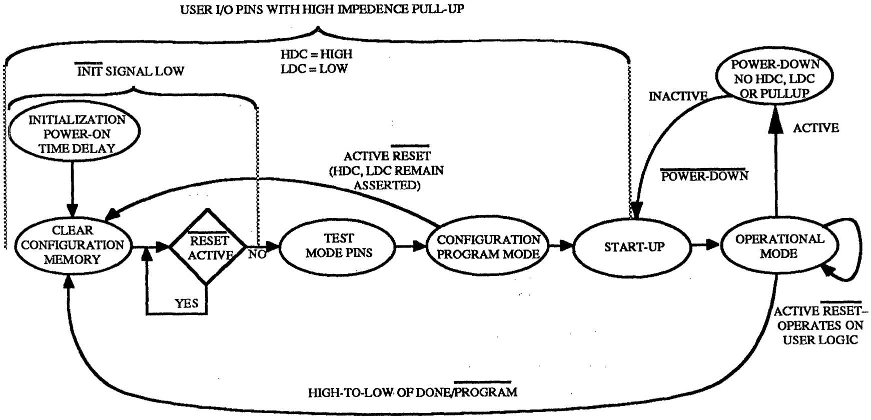
Initialization

An internal power-on reset circuit is triggered when power is applied. When V_{CC} reaches the voltage at which portions of the LCA device begin to operate (2.5 to 3 V), the programmable I/O output buffers are disabled and a high impedance pull-up resistor is provided for the user I/O pins. The programming control I/O pins, D/$\overline{\text{P}}$, HDC and LDC, go active at this time. D/$\overline{\text{P}}$ and LDC remain low and HDC remains high until the end of configuration. A time-out delay is initiated to let the power supply voltage stabilize. During this time, the power-down mode is inhibited. The initialization state time-out (about 11 to 33 ms) is determined by a 14-bit counter driven by a self-generated, internal timer. This nominal 1 MHz timer is subject to variations as a result of process, temperature, and power supply from 0.5 to 1.5 MHz. As shown in the following table, five configuration modes are available, as determined by the input levels of three mode pins M0, M1, and M2.

M0	M1	M2	CLOCK	MODE	DATA
0	0	0	output	Master	Bit Serial
0	0	1	output	Master	Byte Wide (0000 up)
0	1	0	—	—	—
0	1	1	output	Master	Byte Wide (FFFF down)
1	0	0	—	—	—
1	0	1	output	Peripheral	Byte Wide
1	1	0	—	—	—
1	1	1	input	Slave	Bit Serial

Table 1. Mode Selection Table

In master configuration modes, the LCA device becomes the source of the configuration clock (CCLK). The beginning of configuration of devices using peripheral or slave modes must be delayed until they are initialized. An LCA device with mode lines selecting a master configuration mode extends its initialization state using four times the delay (43 to 130 ms). This ensures that all daisy-chained slave devices it may be driving will be ready even if the master is very fast, and the slave(s) very slow. The next figure shows the sequence of states.


USER I/O PINS WITH HIGH IMPEDENCE PULL-UP
$\overline{\text{INIT}}$ SIGNAL LOW
HDC = HIGH
LDC = LOW
INITIALIZATION POWER-ON TIME DELAY
CLEAR CONFIGURATION MEMORY
$\overline{\text{RESET}}$ ACTIVE
NO
YES
TEST MODE PINS
ACTIVE $\overline{\text{RESET}}$ (HDC, LDC REMAIN ASSERTED)
CONFIGURATION PROGRAM MODE
START-UP
OPERATIONAL MODE
POWER-DOWN NO HDC, LDC OR PULLUP
INACTIVE
ACTIVE
$\overline{\text{POWER-DOWN}}$
ACTIVE $\overline{\text{RESET}}$– OPERATES ON USER LOGIC
HIGH-TO-LOW OF DONE/$\overline{\text{PROGRAM}}$

Power-On Delay is

2^{14} Cycles for Non-Master Mode —11 to 33 ms

2^{16} Cycles for Master Mode — 43 to 130 ms

Clear is

~200 Cycles for the Am3020 — 130 to 400 µs

~250 Cycles for the Am3030 --- 165 to 500 µs

~290 Cycles for the Am3042 --- 195 to 580 µs

~330 Cycles for the Am3064 --- 220 to 660 µs

~375 Cycles for the Am3090 — 250 to 750 µs

10642-009A

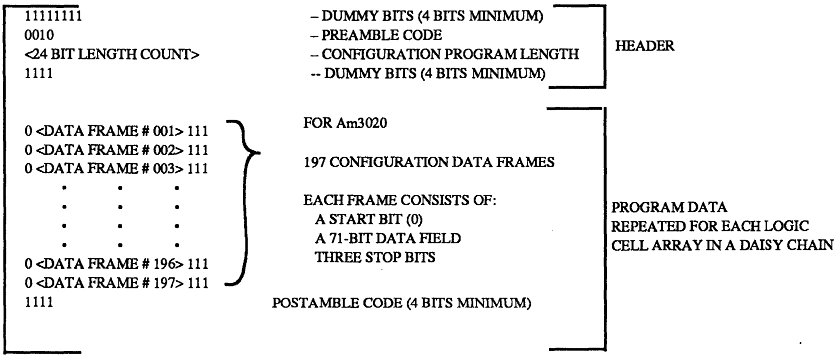
Figure 20. State Diagram of the Configuration Process

At the end of initialization, the LCA device enters the CLEAR state, in which it clears the configuration memory. The active-LOW, open-drain initialization signal, $\overline{\text{INIT}}$, indicates completion of the initialization and clear states by switching HIGH. The LCA device tests for the absence of an external active-LOW $\overline{\text{RESET}}$ before it makes a final sample of the mode lines and enters the configuration state. An external wired-AND of one or more $\overline{\text{INIT}}$ pins can be used to control configuration by the assertion of the active low $\overline{\text{RESET}}$ of a master mode device or to signal a processor that the LCA devices are not yet initialized.

If a configuration has begun, re-asserting $\overline{\text{RESET}}$ for at least three internal timer cycles is recognized, and the LCA device will abort the program, clearing the partially loaded configuration memory words. The LCA device will then re-sample $\overline{\text{RESET}}$ and the mode lines before re-entering the configuration state.

The configuration bit stream is initiated again when a configured LCA device senses a HIGH-to-LOW transition on the DONE/$\overline{\text{PROG}}$ package pin. The LCA device returns to the CLEAR state, in which the configuration memory is cleared and mode lines re-sampled, as for an aborted configuration. The complete configuration bit stream is cleared and loaded during each configuration cycle.

Length count control lets a system of multiple LCA devices, of various sizes, begin synchronized operation. The configuration bit stream generated by the MakePROM software of the PGA Development System begins with a preamble of 111111110010. This is followed by a 24-bit length count representing the total number of configuration clocks needed to complete loading of the configuration program(s). The data framing is shown in the following figure.



11111111 - DUMMY BITS (4 BITS MINIMUM)
0010 - PREAMBLE CODE
<24 BIT LENGTH COUNT> - CONFIGURATION PROGRAM LENGTH
1111 -- DUMMY BITS (4 BITS MINIMUM)
HEADER

0 <DATA FRAME # 001> 111
0 <DATA FRAME # 002> 111
0 <DATA FRAME # 003> 111
. . .
. . .
. . .
. . .
0 <DATA FRAME # 196> 111
0 <DATA FRAME # 197> 111
1111

FOR Am3020

197 CONFIGURATION DATA FRAMES

EACH FRAME CONSISTS OF:
A START BIT (0)
A 71-BIT DATA FIELD
THREE STOP BITS

POSTAMBLE CODE (4 BITS MINIMUM)

PROGRAM DATA
REPEATED FOR EACH LOGIC
CELL ARRAY IN A DAISY CHAIN

Device	Am3020	Am3030	Am3042	Am3064	Am3090
Gates	2000	3000	4200	6400	9000
CLBs Row X Col	64 (8 X 8)	100 (10 X 10)	144 (12 X 12)	224 (16 X 14)	320 (20 X 16)
IOBs	64	80	96	120	144
Flip-flops	256	360	480	688	928
Bits per frame (w/ 1 start 3 stop)	75	92	108	140	172
Frames	197	241	285	329	373
Program Data = Bits * Frames + 4 (excludes preamble)	14779	22176	30784	46064	64160
PROM size (bits)	14816	22216	30824	46104	64200

10642-010A

Figure 21. Internal Configuration Data Structure

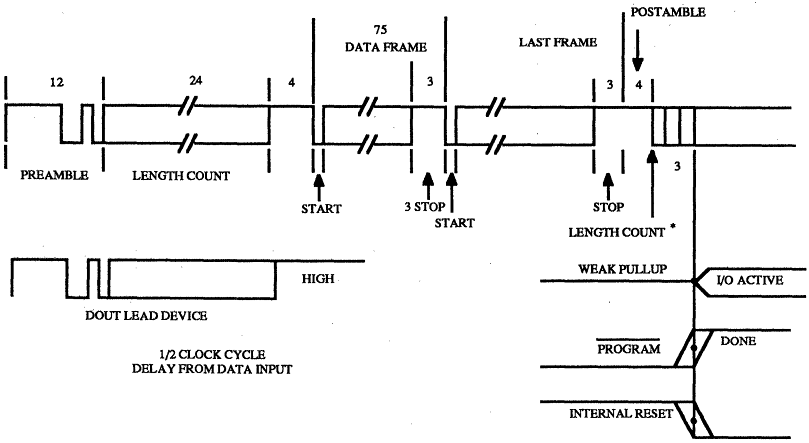
All LCA devices connected in series read and shift preamble and length count in on positive, and out on negative, configuration clock edges. An LCA device that has received the preamble and length count then presents a HIGH Data Out until it has intercepted the appropriate number of data frames. When the configuration memory of an LCA device is full and the length count does not compare, the LCA device shifts any additional data through in the same way it did for preamble and length count.

When the LCA configuration memory is full and the length count compares, the LCA device executes a synchronous start-up sequence and becomes operational, as shown below.

Three CCLK cycles after the completion of loading configuration data, the user I/O pins are enabled as configured. As selected in MakeBits, the internal user-logic reset is released either one clock cycle before, or one clock cycle after the I/O pins become active. A similar timing selection is programmable for the DONE/PROG output signal. DONE/PROG can also be programmed to be an open drain or to include a pull-up resistor to accommodate wired ANDing. High During Configuration (HDC) and Low During Configuration (LDC) are two user I/O pins driven active when an LCA is initializing, clearing, or configuring. These pins and the DONE/PROG commands provide signals for control of external logic signals such as reset, bus enable, or PROM enable during configuration. For parallel master configuration modes, these signals provide PROM enable control and let the data pins be shared with user logic signals.

User I/O inputs can be programmed for either TTL-or CMOS-compatible thresholds. At power-up, all inputs have TTL thresholds and can change to CMOS thresholds at the completion of configuration, if the user has selected CMOS thresholds. The threshold of PWRDWN and the direct clock inputs are fixed at the CMOS level.

If the crystal oscillator is used, it will begin operation before configuration is completed; this allows time for stabilization before the oscillator is connected to the internal circuitry.



*THE CONFIGURATION DATA CONSISTS OF A COMPOSITE 40-BIT PREAMBLE/LENGTH-COUNT, FOLLOWED BY ONE OR MORE CONCATENATED LCA PROGRAMS, SEPARATED BY 4-BIT POSTAMBLES. AN ADDITIONAL FINAL POSTAMBLE BIT IS ADDED FOR EACH SLAVE DEVICE AND THE RESULT ROUNDED UP TO A BYTE BOUNDARY. THE LENGTH COUNT IS TWO LESS THAN THE NUMBER OF RESULTING BITS.

TIMING OF THE ASSERTION OF DONE AND TERMINATION OF THE INTERNAL RESET MAY EACH BE PROGRAMMED TO OCCUR ONE CYCLE BEFORE OR AFTER THE I/O OUTPUTS BECOME ACTIVE.

10642-011A

Figure 22. Configuration and Start-Up

Configuration Data

Configuration data to define the function and interconnection within an LCA device are loaded from an external storage at power-up and on a reprogram signal. Several methods of automatic and controlled loading of the required data are designed into the LCA device. Logic levels applied to mode selection pins at the start of configuration determine the method to be used. See the mode selection Table 1.

The format of the data can be either bit-serial or byte-parallel, depending on the configuration mode. Various AMD programmable gate arrays will have different sizes and numbers of data frames. To maintain compatibility between various device types of the AMD product line, the Am3000 Series LCA devices use formats compatible with the Am2000 Series. For the Am3020, configuration requires 14779 bits, arranged in 197 data frames, for each device. An additional 40 bits are used in header, as shown in the previous figure. The specific data format for each device is produced by the MakeBits command of the PGA Development System.

One or more of these files can then be combined and appended to a length count preamble and be transformed into a PROM format file by the MakePROM command of the PGA Development System. An exception to the compatibility of the devices is that an Am2000 Series device cannot be used as the master for Am3000 Series devices if the Am3000 device has DONE or RESET programmed to occur after their outputs become active. The TIE option of MakeBits defines output levels of unused blocks of a design and connects these to unused routing resources. This prevents indeterminate levels that might produce parasitic supply currents. If unused blocks are not sufficient to compute the tie, the FLAGNET command can be used to indicate nets that must not be used to drive the remaining unused routing, as that might affect timing of user nets. NORESTORE will retain the results of TIE for timing analysis with QUERYNET, before RESTORE returns the design to the untied condition. TIE can be omitted for quick breadboard iterations where a few additional milliamps of I_{CC} are acceptable.

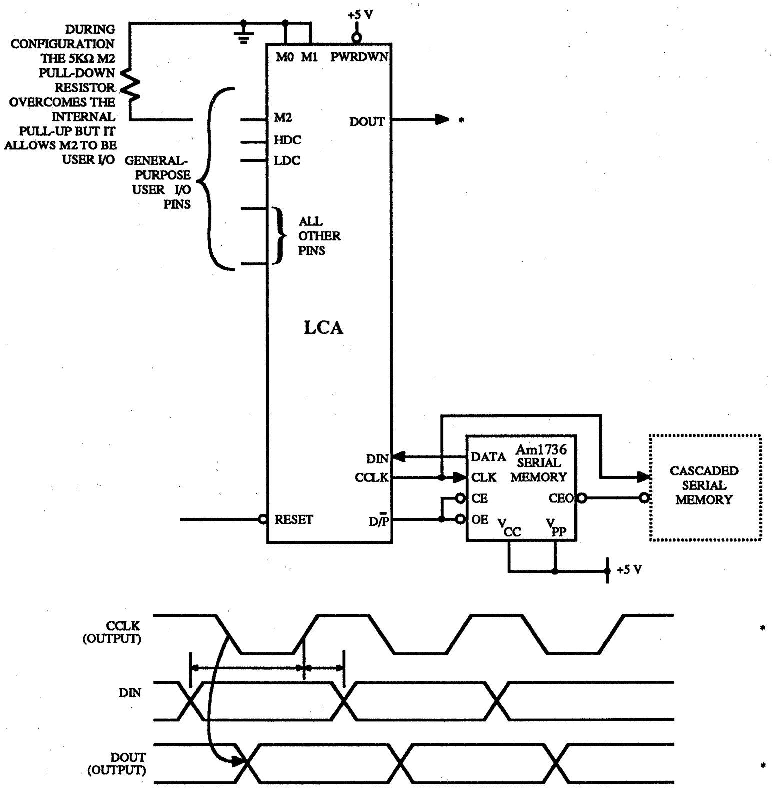
The configuration bit stream begins with HIGH preamble bits, a 4-bit preamble code, and a 24-bit length count. When configuration is initiated, a counter in the LCA device is set to 0 and begins to count the total number of configuration clock cycles applied to the device. As each configuration data frame is supplied to the LCA device, it is internally assembled into a data word. As each data word is completely assembled, it is loaded in parallel into one word of the internal configuration memory array. The configuration loading process is completed when the current length count equals the loaded length count, and the required configuration bit stream data frames have been written. Internal user flip-flops are held reset during configuration.

Two user-programmable pins are defined in the unconfigured LCA device. HDC and LDC, as well as DONE/PROG, can be used as external control signals during configuration. In master mode configurations it is convenient to use LDC as an active-LOW EPROM Chip Enable. After the last configuration data-bit is loaded and the length count compares, the user I/O pins become active. Options in the MakeBits software allow timing choices of one clock earlier or later for the timing of the end of the internal logic reset and the assertion of the DONE signal. The open-drain DONE/PROG output can be AND-tied with multiple LCA devices and used as an active-HIGH READY, an active-LOW PROM enable, or a RESET to other portions of the system.

Master Mode

In master mode, the LCA device automatically loads configuration data from an external memory device. There are three master modes that use the internal timing source to time the incoming data supplying the configuration clock (CCLK). Serial master mode uses serial configuration data supplied to data-in (DIN) from a synchronous serial source such as the AMD Serial Configuration PROM (Am1736) as shown in Figure 23.

The one-time-programmable Am1736 Serial Configuration PROM supports automatic loading of configuration programs up to 36K bits. Multiple devices can be cascaded to support additional LCA devices. An early DONE inhibits the Am1736 data output one CCLK cycle before the LCA I/O becomes active.



*FOR OPTIONAL SLAVE MODE LCAs IN A DAISY CHAIN

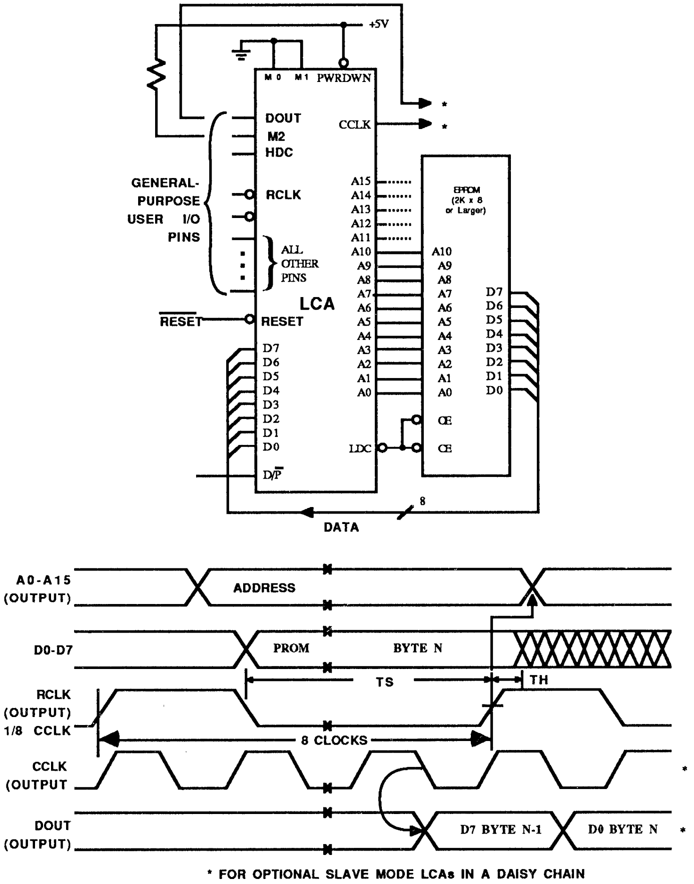
10642-012A

Figure 23. Master Serial Mode

Parallel master LOW and master HIGH modes automatically use parallel data supplied to the D0–D7 pins in response to the 16-bit address generated by the LCA device. Figure 24 shows an example of the parallel master mode connections required. The LCA HEX starting address is 0000 and increments for master LOW mode. It is FFFF and decrements for master HIGH mode. These two modes provide address compatibility with microprocessors beginning execution from opposite ends of memory.

For master HIGH or LOW, data bytes are read in parallel by each read clock output pulse (RCLK) and internally serialized by the configuration clock. As each data byte is read, the least-significant bit of the next byte, D0, becomes the next bit in the internal serial configuration word. One master mode LCA device can be used to interface the configuration program-store and pass additional concatenated configuration data to additional LCA devices in a serial daisy-chain fashion. CCLK is provided for the slaved devices and their serialized data is supplied from DOUT to DIN – DOUT to DIN, etc.

Configuration data are loaded automatically from an external byte wide PROM. An early DONE inhibits the PROM outputs a CCLK before the LCA I/O becomes active.



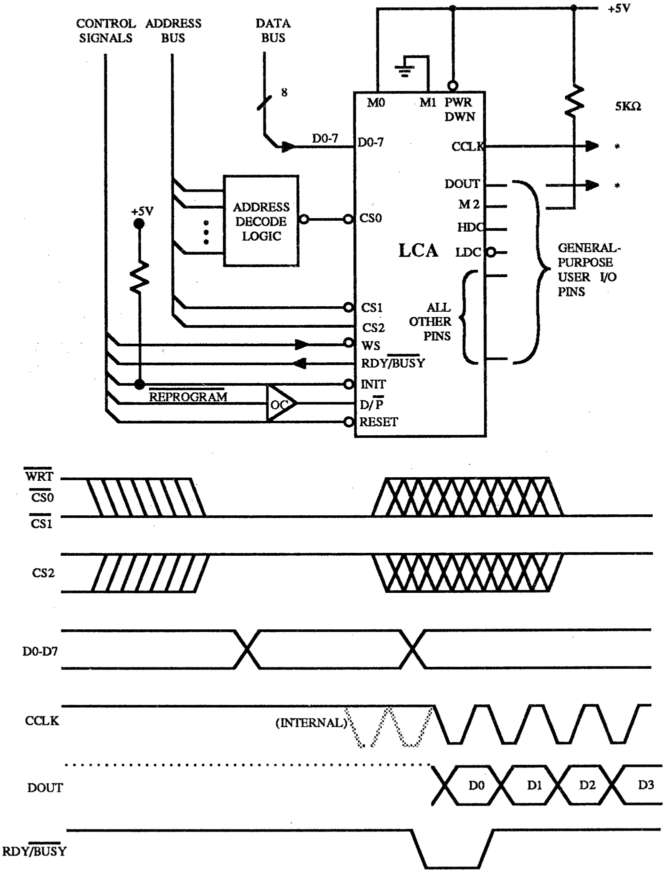
* FOR OPTIONAL SLAVE MODE LCAs IN A DAISY CHAIN

Figure 24. Master Parallel Mode

Peripheral Mode

Peripheral mode provides a simplified interface through which the device can be loaded byte-wide, as a processor peripheral. The next figure shows the peripheral mode connections. Processor write cycles are decoded from the common assertion of the active-LOW Write Strobe ($\overline{\text{WRT}}$), and two active-LOW and one active HIGH Chip Selects (CS0, CS1, CS2). If all these signals are not available, the unused inputs should be driven to their respective active levels. The LCA device accepts one byte of configuration data on the D0–D7 inputs for each selected processor write cycle. Each byte of data is loaded into a buffer register.

The LCA device generates a configuration clock from the internal timing generator and serializes the parallel input data for internal framing or for succeeding slaves on Data Out (DOUT). An output HIGH on the READY/BUSY pin indicates completion of loading for each byte and that the input register is ready for a new byte. As with master modes, peripheral mode can also be used as a lead device for a daisy-chain of slave devices.



*FOR OPTIONAL SLAVE MODE LCAs IN A DAISY CHAIN

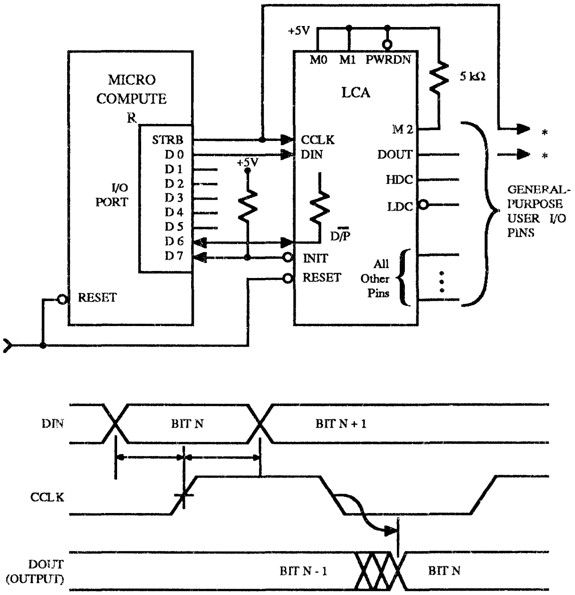
10642-013A

Figure 25. Peripheral Mode

Slave Mode

Slave mode provides a simple interface for loading the LCA device configuration, as shown below. Serial data are supplied in conjunction with a synchronizing input clock. Bit-serial configuration data are read at rising edge of the CCLK. Data on DOUT are provided on the falling edge of CCLK.

Most slave mode applications are in daisy-chain configurations in which the data input are supplied by the previous LCA's data out, while the clock is supplied by a lead device in master or peripheral mode. Data can also be supplied by a processor or other special circuits.



* FOR OPTIONAL SLAVE MODE LCAs IN A DAISY CHAIN

10642-014A

Figure 26. Slave Mode

Daisy-Chain

The AMD PGA Development System is used to create a composite configuration bit stream for selected LCA devices. This configuration includes the following:

- A preamble
- A length count for the total bit stream
- Multiple concatenated data programs
- A postamble
- An additional fill bit per device in the serial chain

After loading and passing on the preamble and length count to a possible daisy-chain, a lead device will load its configuration data frames while providing a HIGH DOUT to possible down-stream devices as shown below. In this figure, all are configured from the common EPROM source. The slave mode device $\overline{\text{INIT}}$ signals delay the master device configuration until they are initialized. A well defined termination of SYSTEM RESET is needed when controlling multiple LCA devices.

Loading continues until the current length count has reached the full value. The additional data are passed through the lead device and appear on the DOUT pin in serial form. The lead device also generates the CCLK to synchronize the serial output data and data in of LCA devices further attached. Data are read in on DIN of slave devices by the positive edge of CCLK and shifted out the DOUT on the negative edge of CCLK. A parallel master mode device uses its internal timing generator to produce an internal CCLK of eight times its EPROM address rate, while a peripheral mode device produces a burst of eight CCLKs for each chip select and write-strobe cycle. The internal timing generator continues to operate for general timing and synchronization of inputs in all modes.

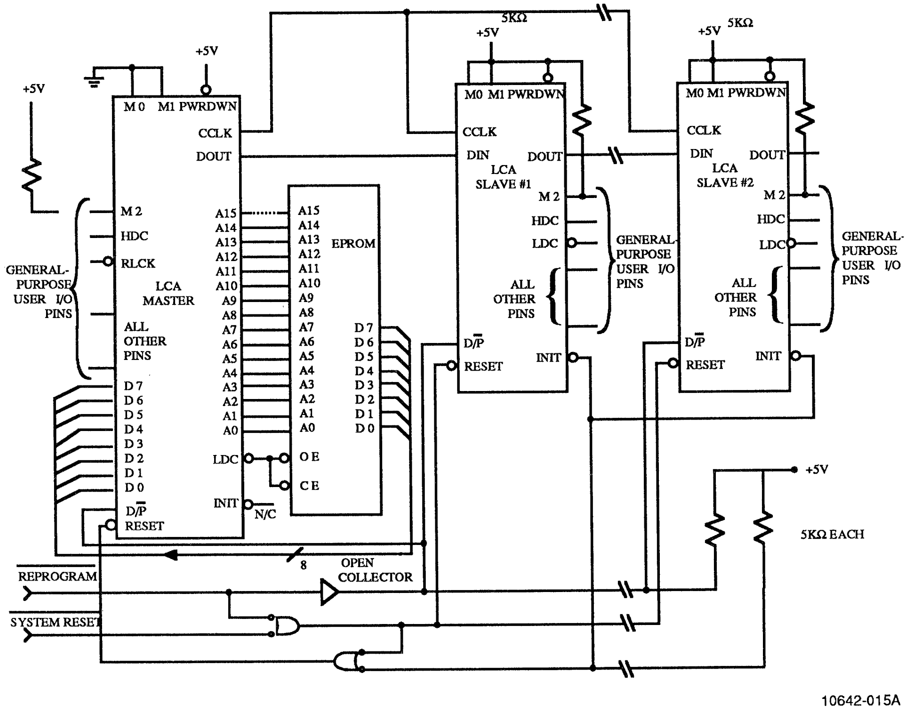


Figure 27. Master Mode Configuration with Daisy Chained Slave Mode Devices

SPECIAL CONFIGURATION FUNCTIONS

The configuration data include control over several special functions, in addition to the normal user logic functions and interconnections.

- Input thresholds
- Readback enable
- DONE pull-up resistor
- DONE timing
- RESET timing
- Oscillator frequency divided-by-two

Each of these functions is controlled by configuration data bits selected as part of the normal PGA Development System bit-stream generation process.

Input Thresholds

Prior to the completion of configuration, all LCA input thresholds are TTL compatible. Upon completion of configuration, the input thresholds become either TTL or CMOS compatible, as programmed. The use of the TTL threshold option requires some additional supply current for threshold shifting. The exception is the threshold of the $\overline{\text{PWRDWN}}$ input and direct clocks, which always have a CMOS input. Prior to the completion of configuration, the user I/O pins have a high-impedance pull-up. The configuration bit stream can be used to enable the IOB pull-up resistors in the operational mode to act either as an input load or to avoid a floating input on an otherwise unused pin.

Readback

The contents of an LCA device can be read back if it has been programmed with a bit stream in which the Readback option has been enabled. Readback can be used for verification of configuration, as well as a method of determining the state of internal logic nodes during debugging with the In-Circuit debugger. There are three options in generating the configuration bit stream:

- **Never** inhibits the Readback capability.
- **One-time** inhibits Readback after one Readback has been executed to verify the configuration.
- **On-command** permits unrestricted use of Readback.

Readback is done without the use of any of the user I/O pins; only M0, M1, and CCLK are used. The initiation of readback is produced by a LOW-to-HIGH transition of the M0/RTRIG (Read Trigger) pin. Once the READBACK command has been given, the input CCLK is driven by external logic to read back each data bit in a format similar to loading. After two dummy bits, the first data frame is shifted out, in inverted sense, on the M1/$\overline{\text{RDATA}}$ (Read Data) pin. All data frames must be read back to complete the process and return the mode select and CCLK pins to their normal functions.

The readback data includes the current state of each internal logic block storage element, and the state of the .i and .ri connection pins on each IOB. These data are imbedded into unused configuration bit positions during readback. This state information is used by the PGA Development System In-Circuit Verifier to provide visibility into the internal operation of the logic while the system is operating. To read back a uniform time-sample of all storage elements it may be necessary to inhibit the system clock.

Reprogram

The LCA configuration memory can be re-written while the device is operating in the user's system. To initiate a reprogramming cycle, the dual function package pin DONE/$\overline{\text{PROG}}$ must transition from HIGH to LOW. To reduce noise sensitivity, the input signal is filtered for two cycles of the LCA's internal timing generator. When reprogram begins, the user-programmable I/O output buffers are disabled, and high-impedance pull-ups are provided for the package pins. The device returns to the CLEAR state and clears the configuration memory before it is initialized.

Reprogram control is often exercised using an external open-collector driver that pulls DONE/$\overline{\text{PROG}}$ LOW. Once it recognizes a stable request, the LCA device holds a LOW until the new configuration has been completed. Even if the reprogram request is externally held LOW beyond the configuration period, the LCA device will begin operation upon completion of configuration.

DONE Pull-up

DONE/$\overline{\text{PROG}}$ is an open-drain I/O pin indicating that the LCA device is operational. An optional internal pull-up resistor can be enabled by the user of the PGA Development System when MakeBits is executed. The DONE/$\overline{\text{PROG}}$ pins of multiple LCA devices in a daisy-chain can be connected to indicate all are DONE or to direct them to reprogram.

DONE Timing

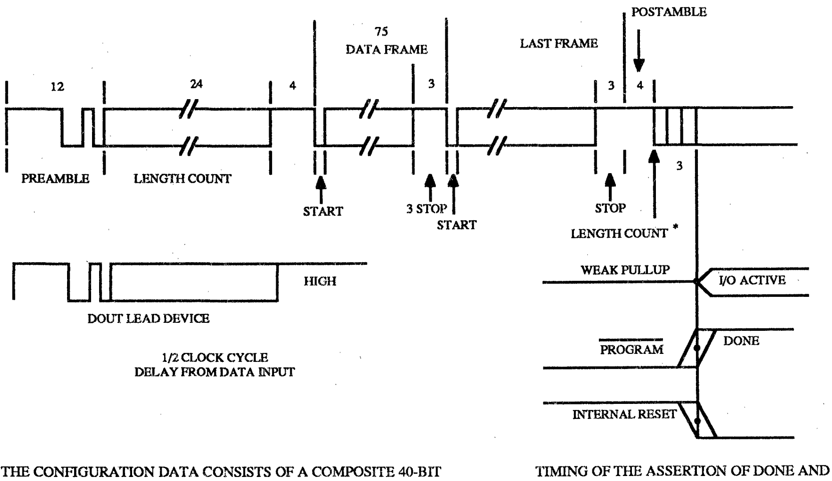
By a selection in the MakeBits program, the timing of the DONE status signal can be controlled to occur one CCLK cycle before, or one cycle after, the timing of outputs are activated. This is shown below. This facilitates control of external functions such as a PROM enable or holding a system in a wait state.

RESET Timing

As with DONE timing, the timing of the release of the internal RESET can be controlled by a selection in the MakeBits program. It then occurs one CCLK cycle before, or one cycle after, the timing of outputs are enabled, as shown above. This reset maintains all user-programmable flip-flops and latches in a zero state during configuration.

Crystal Oscillator Division

A selection in the MakeBits software lets the user incorporate a dedicated divide-by-two flip-flop in the crystal oscillator function. This helps ensure a symmetrical timing signal. Although the frequency stability of crystal oscillators is high, the symmetry of the waveform can be affected by bias or feedback drive.



*THE CONFIGURATION DATA CONSISTS OF A COMPOSITE 40-BIT PREAMBLE/LENGTH-COUNT, FOLLOWED BY ONE OR MORE CONCATENATED LCA PROGRAMS, SEPARATED BY 4-BIT POSTAMBLES. AN ADDITIONAL FINAL POSTAMBLE BIT IS ADDED FOR EACH SLAVE DEVICE AND THE RESULT ROUNDED UP TO A BYTE BOUNDARY. THE LENGTH COUNT IS TWO LESS THAN THE NUMBER OF RESULTING BITS.

TIMING OF THE ASSERTION OF DONE AND TERMINATION OF THE INTERNAL RESET MAY EACH BE PROGRAMMED TO OCCUR ONE CYCLE BEFORE OR AFTER THE I/O OUTPUTS BECOME ACTIVE.

10642-016A

Figure 28. Configuration and Start-Up

PERFORMANCE

Device Performance

The high performance of the LCA device is due in part to the manufacturing process, which is similar to that used for other high-speed CMOS logic devices. Performance can be measured in terms of minimum propagation times for logic elements. One parameter traditionally used to describe the overall performance of a gate array is the toggle frequency of a flip-flop. The configuration for determining the toggle performance of the LCA device is shown at right. The flip-flop output Q is fed back through the combinational logic as $\overline{Q}$ to form the toggle flip-flop.

Actual LCA device performance is determined by the timing of critical paths, including both the fixed timing for the logic and storage elements in that path, and the timing associated with the network routing.

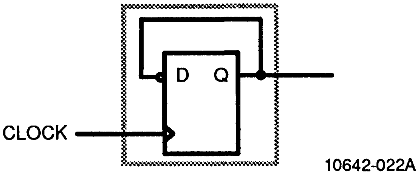
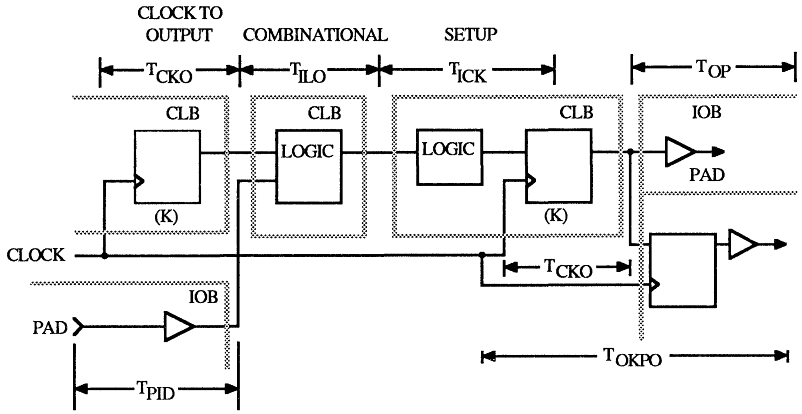


Figure 29. "Toggle" Flip-Flop

Examples of internal worst-case timing are included in the performance data to allow the user to make the best use of the device's capabilities. The PGA Development System timing calculator, or LCA-generated simulation models, should be used to calculate worst-case paths by using actual impedance and loading information. The following figure shows a variety of elements used involved in determining system performance.



			MIL/COM'L		MIL/COM'L		COM'L		Unit
		Speed Grade	–50		–70		–100		
	Description	Symbol	Min.	Max.	Min.	Max.	Min.	Max.	
Logic input to Output	Combinational	T_{ILO}		14		9		7	ns
K Clock	To output	T_{CKO}		12		8		7	ns
	Logic-in put setup	T_{ICK}	12		8		7		ns
	Logic-in put hold	T_{CKI}	0		0		0		ns
Input/Output	Pad to input (direct)	T_{PID}		10		7		4	ns
	Output to pad (enabled fast)	T_{OPF}		14		10		6	ns
	I/O clock to pad	T_{OKPO}		18		13		10	ns
FF toggle frequency		F_{CLK}		50		70		100	MHz

Figure 30. Examples of Primary Block Speed Factors

The speed of internal elements is determined by differential measurements of package pins. The performance of a user's design can be predicted by the PGA Development System delay calculator.

Actual measurement of internal timing is not practical; often only the sum of component timing is relevant, as in the case of input to output. The relationship between input and output timing is arbitrary; only the total determines performance. Timing components of internal functions can be determined by measuring the differences at the pins of the package. A synchronous logic function with a clock-to-block output, and a block-input to clock set-up, is capable of higher speed operation than a logic configuration of two synchronous blocks with an extra combinational block level between them. System clock rates to 60% of the toggle frequency are practical for logic in which there is an extra combinational level located between synchronized blocks. This permits implementation of functions of up to 25 variables. The use of the wired-AND is also available for wide, high-speed functions.

Logic Block Performance

Logic block performance is expressed as the propagation time from the interconnection point at the input of the combinational logic to the output of the block in the interconnection area. Combinational performance is independent of the specific logic function, which is based on look-up tables.

The only parameter for all logic functions is the Logic Input to Output delay. For combinational logic used in conjunction with the storage element, however, there are two critical parameters. First, for the combinational logic function driving the data input of the storage element, the critical timing is data setup relative to the clock edge provided to the flip-flop. Second, for the signals then produced by the storage elements, the critical timing is the Clock to Output delay. These parameters are shown in the previous figure.

Logic block loading is limited only by the user's propagation delay requirements. Speed performance of the logic block is a function of the supply voltage and the temperature, as shown in the next two figures.

The following figure shows the change in speed performance as a function of temperature. The variation is normalized for 30° C, 70° C, 85° C, and 125° C.

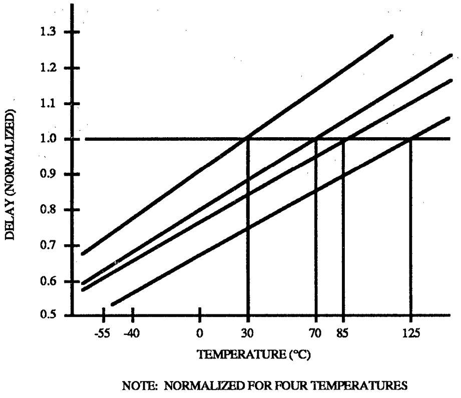


Figure 31. Delay as a Function of Temperature

The following figure shows how the speed performance of a CMOS device increases with V_{CC} within the operating range.

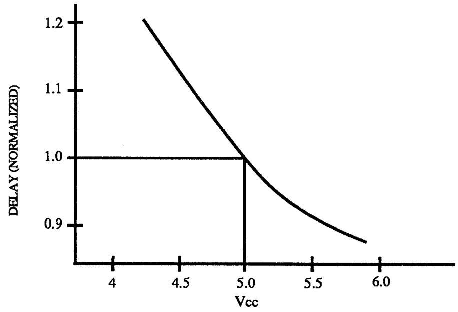


Figure 32. Delay as a Function of V_{CC}

Interconnection Performance

Interconnection performance depends on the routing resource used for the signal path. As discussed earlier, direct interconnection from block to block provides a fast path for a signal. The single metal segment used for long lines exhibits low resistance from end to end, but relatively high capacitance. Signals driven through a programmable switch will have the additional impedance of the switch added to their normal drive impedance.

General-purpose interconnection performance depends on the number of switches and segments used, the presence of the bidirectional re-powering buffers, and the loading at all points on the signal path. In calculating the worst-case timing for a general interconnection path, the timing calculator portion of the PGA Development System takes all of these elements into account.

As an approximation, interconnection timing is proportional to the summation of totals of local metal segments beyond each programmable switch. In effect, the time is a sum of R–C time, each approximated by an R times the total C it drives. The R of the switch and the C of the interconnection are functions of the particular device performance grade.

For a string of three local interconnections, the approximate time at the first segment (after the first switch resistance) would be three units; after the next switch there are an additional two units; and there is an additional unit after the last switch in the chain. The interconnection R–C chain terminates at each re-powering buffer. The capacitance of the block inputs is not significant; the capacitance is in the interconnection metal and switches, as shown in the following figure.

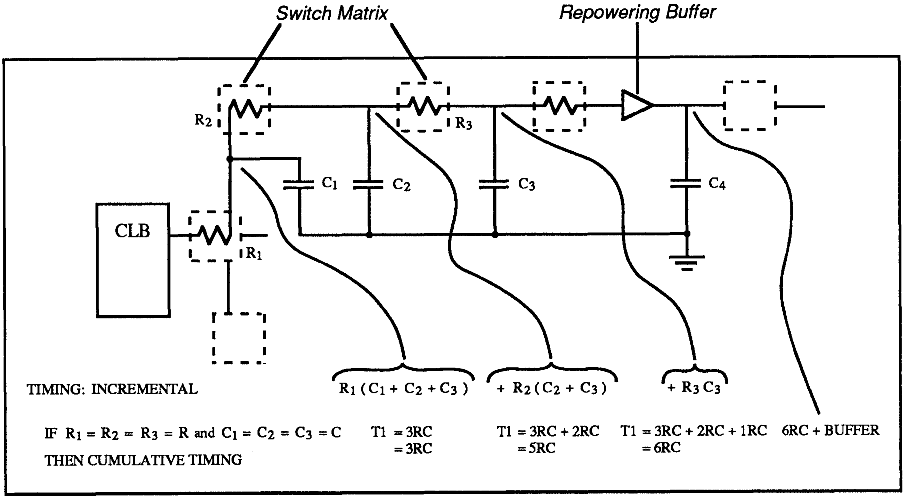


Figure 33. Interconnection Timing Example

POWER

Power Distribution

Power for the LCA device is distributed through a grid to achieve high noise immunity and isolation between the logic and I/O. Inside the LCA device, a dedicated V_{CC} and ground ring surrounding the logic array provide power to the I/O drivers, as shown below. An independent matrix of V_{CC} and ground lines supplies the interior logic of the device. This power distribution grid provides a stable supply and ground for all internal logic; this assumes that the external package power pins are all connected and appropriately decoupled. Usually, a 0.1-µF capacitor connected near the V_{CC} and ground pins of the package will provide adequate decoupling.

Output buffers capable of driving the specified 4-mA loads under worst-case conditions can be capable of driving 25 to 30 times that current in a best case. Noise can be reduced by minimizing external load capacitance and reducing simultaneous output transitions in the same direction. Also, it may be beneficial to locate heavily loaded output buffers near the ground pads. The IOB output buffers have a slew-limited mode that should be used where output rise and fall times are not speed critical. A lower AC drive current reduces transition and supply noise without a corresponding reduction in DC drive. Slew-limited outputs maintain their DC drive capability, but generate less external reflections and internal noise.

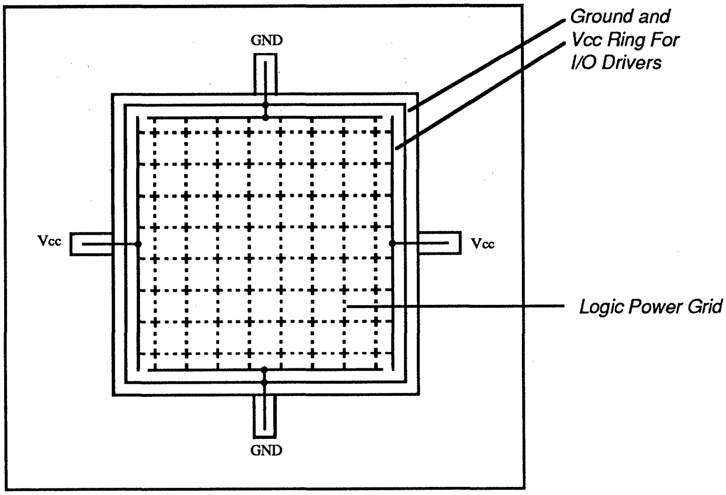
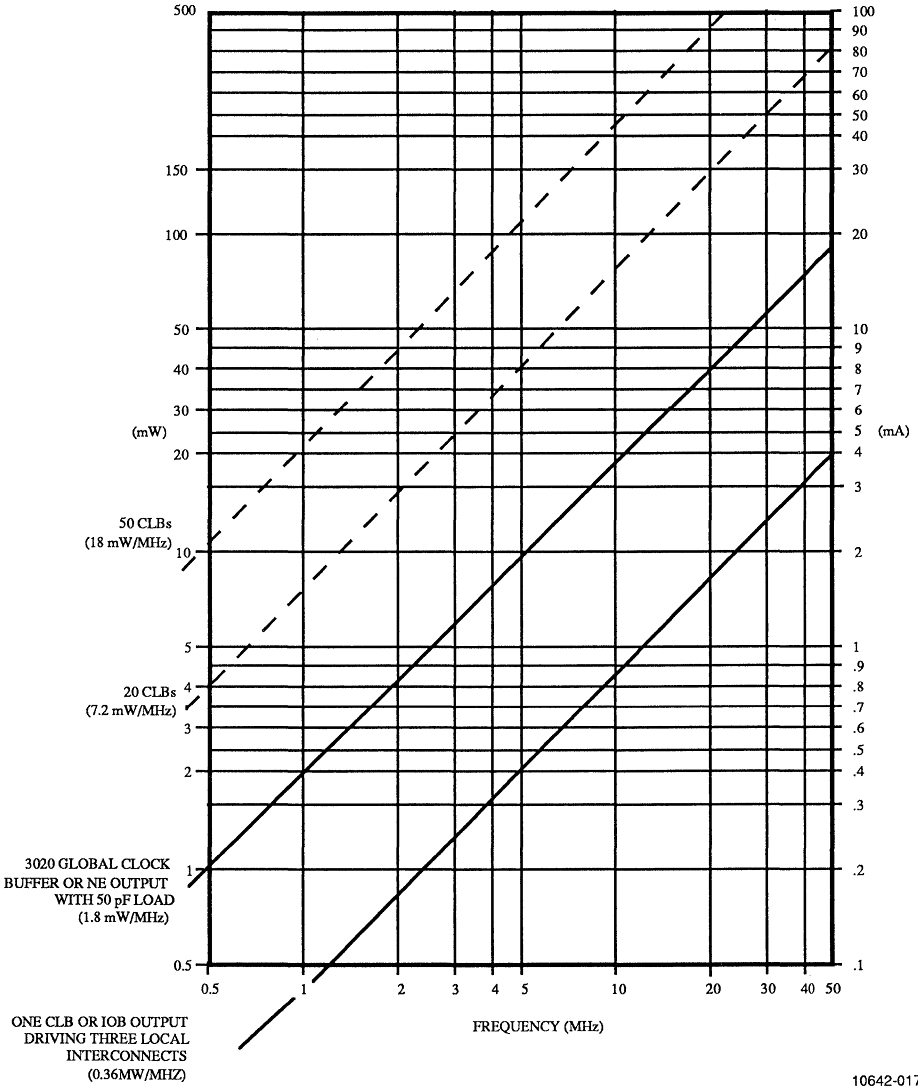


Figure 34. LCA Power Distribution

Power Dissipation

The LCA device exhibits the low power consumption characteristic of CMOS ICs. For any design, the user can use the figure below to calculate the total power requirement based on the sum of the external and internal capacitive and DC loads. The total chip power is the sum of Vcc • ICCO, plus internal and external values of capacitive charging currents and resistive loads.

The configuration options of TTL chip input threshold requires power for the threshold reference. The power required by the static memory cells holding the configuration data is very low and can be maintained in a power-down mode.



10642-017A

Figure 35. LCA Power Consumption by Element

Usually, most power dissipation is produced by external capacitive loads on the output buffers. The load- and frequency-dependent power is 25 µW/pF/MHz per output. Another component of I/O power is the DC loading on each output pin by LCA-driven devices.

Internal power dissipation is a function of the number and size of the nodes, and the frequency at which they change. In an LCA device, the fraction of nodes changing on a given clock is typically low (10–20). For example, in a large binary counter, the average clock cycle produces changes equal to one CLB output at the clock frequency. Typical global clock buffer power is between 1.7 mV/MHz for the Am3020 and 3.6 mV/MHz for the Am3090. The internal capacitive load is more a function of interconnection than fanout. With typical load of three general interconnection segments, each CLB output requires about 0.4 mW/MHz of its output frequency.

Total power = V_{CC} • ICCO + external (DC + capacitive) + internal (CLB + IOB + Long Line + pull-up)

Because the control storage of the LCA device is CMOS static memory, its cells require a very low standby current for data retention. In some systems, this characteristic can be used as a method of preserving configurations in the event of a primary power loss. The LCA device has built in power-down logic that, when activated, will disable normal operation of the device and retain only the configuration data. All internal operation is suspended and output buffers are placed in a high impedance state with no pull-ups. Power-down data retention is possible with a simple battery-backup circuit because the power requirement is extremely low. For retention at 2.4 V, the required current is typically on the order of 50 nA.

To force the LCA device into the power-down state, the user must pull the $\overline{\text{PWRDWN}}$ pin LOW and continue to supply a retention voltage to the V_{CC} pins of the package. When normal power is restored, V_{CC} is increased to its normal operating voltage and $\overline{\text{PWRDWN}}$ is returned to HIGH. The LCA device resumes operation with the same internal sequence that occurs at the conclusion of configuration. Internal I/O and logic block storage elements will be reset, the outputs will become enabled, and the DONE/$\overline{\text{PROG}}$ pin will be released. No configuration programming is required.

When the power supply is removed from a CMOS device, it is possible to supply some power from an input signal. The conventional electrostatic input protection is provided by diodes to the supply and ground. A positive voltage applied to an input or output will cause the positive protection diode to conduct and drive the power pin. This condition can produce invalid power conditions and should be avoided. A large series resistor can be used to limit the current, or a bipolar buffer can be used to isolate the input signal.

PIN DESCRIPTIONS

Permanently Dedicated Pins

V_{CC}

Two to eight (depending on package type) connections to the nominal +5 V supply voltage. All must be connected.

GND

Two to eight (depending on package type) connections to ground. All must be connected.

PWRDWN

An active-LOW power-down input stops all internal activity to minimize V_{CC} power and puts all output buffers in a high-impedance state. Configuration is retained, however internal storage elements are reset. When the PWRDWN pin returns HIGH, the device returns to operation with the same sequence of reset, buffer enable, and DONE/PROG as at the completion of configuration.

RESET

This active LOW input has three functions. Prior to the start of configuration, a LOW input will delay the start of configuration. An internal circuit senses the application of power and begins a minimal time-out cycle. When the time-out and RESET are complete, the levels of the M lines are sampled and configuration begins. If RESET is asserted during a configuration, the LCA device is re-initialized and will restart the configuration at the termination of RESET. If RESET is asserted after configuration is complete, it will provide an asynchronous reset of all IOB and CLB storage elements of the LCA device.

CCLK

During configuration, Configuration Clock is an output of an LCA device in master mode or peripheral mode. LCA devices in slave mode use it as a clock input. During a Readback operation, it is a clock input for the configuration data being shifted out.

DONE

The DONE output is configurable as an open drain with or without a pull-up resistor. At the completion of configuration, the circuitry of the LCA device becomes active in a synchronous order, and DONE can be programmed to occur one cycle before or after.

or

PROG

Once configuration is completed, a HIGH-to-LOW transition of this pin will cause an initialization of the LCA device and start a reconfiguration.

M0

As Mode 0, this input, M1, and M2 are sampled before the start of configuration to establish the configuration mode to be used.

or

RTRIG

As a Read Trigger, after configuration is complete, an input transition to a HIGH will initiate a Readback of configuration and storage element data by CCLK. This operation can be limited to a single request, or can be inhibited altogether, by selecting the appropriate readback option when generating the bit stream.

M1

As Mode 1, this input, M0, and M2 are sampled before the start of configuration to establish the configuration mode. If readback is to be used, a 5-kΩ resistor should be used to define mode level inputs.

or

RDATA

As an active LOW Read Data, this pin is the output of the readback data after configuration is complete.

User I/O Pins That Can Have Special Functions

M2

As Mode 2 this input has a passive pullup during configuration. Together with M0 and M1, it is sampled before the start of configuration to establish the configuration mode. After configuration, this pin becomes a user-programmable I/O pin.

HDC

High During Configuration is held at a HIGH level by the LCA device until after configuration. It is available as a control indication that configuration is not completed. After configuration, this pin is a user I/O pin.

LDC

Low During Configuration is held at a LOW level by the LCA device until after configuration. It is intended to be available as a control indication that configuration is not completed. It is particularly useful in master mode as a LOW enable for an EPROM. After configuration, this pin is a user I/O pin. If used as a LOW EPROM enable, it must be programmed as a HIGH after configuration.

$\overline{\text{INIT}}$

This is an active-LOW-open drain output that is held LOW during the power stabilization and internal clearing of the configuration memory. It can be used to indicate status to a configuring microprocessor, or as a wired-AND of several slave mode devices, a hold-off signal for a master mode device. After configuration, this pin becomes a user-programmable I/O pin.

BCLKIN

This is a direct CMOS level input to the alternate clock buffer (auxiliary buffer) in the lower right corner.

or

XTL1

This user I/O pin can be configured to operate as the output of an amplifier driving an external crystal and bias circuitry.

XTL2

This user I/O pin can be configured to operate as the input of an amplifier usable with an external crystal and bias circuity. This I/O block is left unconfigured. The oscillator configuration is activated by routing a net from the oscillator buffer symbol output and by the MakeBits program.

$\overline{\text{CS0}}$, $\overline{\text{CS1}}$, CS2, $\overline{\text{WRT}}$

These four inputs represent a set of signals, three active LOW and one active HIGH, which are used in peripheral mode to control configuration data entry. The assertion of all four generates a write to the internal data buffer. The removal of any assertion results in the present data of D0–D7 being clocked in.

$\overline{\text{RCLK}}$

During master parallel mode configuration, this pin represents a read of an external memory device.

RDY/$\overline{\text{BUSY}}$

During peripheral parallel mode configuration, this pin indicates when the chip is ready for another byte of data to be written to it. After configuration is complete, this pin becomes a user-programmed I/O pin.

D0–D7

This set of eight pins represents the parallel configuration byte for the parallel master and peripheral modes. After configuration is complete, they are user-programmable I/O pins.

A0–A15

This set of 16 pins presents an address output for a configuration EPROM during master parallel mode. After configuration is complete, they are user-programmable I/O pins.

DIN

This user I/O pin is used as serial Data In during slave or master serial configuration. This pin is Data 0 input in master or peripheral configuration mode.

DOUT

This user I/O pin is used during configuration to output serial configuration data for the Data In of daisy-chained slaves.

TCLKIN

This is a direct CMOS level input to the global clock buffer.

Unrestricted User I/O Pins

I/O

A pin that, after configuration, can be programmed by the user to be an input and/or output pin. Some of these pins present a high-impedance pull-up or perform other functions before configuration is complete.

APPENDIX A: TABLES AND DIAGRAMS

Table 2a. Am3000 Family Configuration Pin Assignments

CONFIGURATION MODE: <M2:M1:M0>					68 PLCC	84 PLCC	84 PGA	132 PGA	175 PGA	USER OPERATION
SLAVE <1:1:1>	MASTER-SER <0:0:0:>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>						
$\overline{\text{PWRDWN}}$ (I)	$\overline{\text{PWRDWN}}$ (I)	$\overline{\text{PWRDWN}}$ (I)	$\overline{\text{PWRDWN}}$ (I)	$\overline{\text{PWRDWN}}$ (I)	10	12	B2	A1	B2	$\overline{\text{PWRDWN}}$ (I)
Vcc	Vcc	Vcc	Vcc	Vcc	18	22	F3	C8	D9	Vcc
M1 (HIGH) (I)	M1 (LOW) (I)	M1 (LOW) (I)	M1 (HIGH) (I)	M1 (LOW) (I)	25	31	J2	B13	B14	$\overline{\text{RDATA}}$
M0 (HIGH) (I)	M0 (LOW) (I)	M0 (HIGH) (I)	M0 (LOW) (I)	M0 (LOW) (I)	26	32	L1	A14	B15	RTRIG (I)
M2 (HIGH) (I)	M2 (LOW) (I)	M2 (HIGH) (I)	M2 (HIGH) (I)	M2 (HIGH) (I)	27	33	K2	C13	C15	I/O
HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	HDC (HIGH)	28	34	K3	B14	E14	I/O
$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	$\overline{\text{LDC}}$ (LOW)	30	36	L3	D14	D16	I/O
$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	$\overline{\text{INIT}}$	34	42	K6	G14	H15	I/O
GND	GND	GND	GND	GND	35	43	J6	H12	J14	GND
					43	53	L11	M13	P15	XTL2 OR I/O
$\overline{\text{RESET}}$ (I)	$\overline{\text{RESET}}$ (I)	$\overline{\text{RESET}}$ (I)	$\overline{\text{RESET}}$ (I)	$\overline{\text{RESET}}$ (I)	44	54	K10	P14	R15	$\overline{\text{RESET}}$ (I)
DONE	DONE	DONE	DONE	DONE	45	55	J10	N13	R14	$\overline{\text{PROG}}$ (I)
		DATA 7 (I)	DATA 7 (I)	DATA 7 (I)	46	56	K11	M12	N13	I/O
					47	57	J11	P13	T14	XTL1 OR I/O
		DATA 6 (I)	DATA 6 (I)	DATA 6 (I)	48	58	H10	N11	P12	I/O
		DATA 5 (I)	DATA 5 (I)	DATA 5 (I)	49	60	F10	M9	T11	I/O
		CS0 (1)			50	61	G10	N9	R10	I/O
		DATA 4 (I)	DATA 4 (I)	DATA (4)	51	62	G11	N8	R9	I/O
Vcc	Vcc	Vcc	Vcc	Vcc	52	64	F9	M8	N9	Vcc
		DATA 3 (I)	DATA 3 (I)	DATA 3 (I)	53	65	F11	N7	P8	I/O
		$\overline{\text{CS1}}$ (I)			54	66	E11	P6	R8	I/O
		DATA 2 (I)	DATA 2 (I)	DATA 2 (I)	55	67	E10	M6	R7	I/O
		DATA 1 (I)	DATA 1 (I)	DATA 1 (I)	56	70	D10	M5	R5	I/O
		RDY/$\overline{\text{BUSY}}$	$\overline{\text{RCLK}}$	$\overline{\text{RCLK}}$	57	71	C11	N4	P5	I/O
DIN (I)	DIN (I)	DATA 0 (I)	DATA 0 (I)	DATA 0 (I)	58	72	B11	N2	R3	I/O
DOUT	DOUT	DOUT	DOUT	DOUT	59	73	C10	M3	N4	I/O
CCLK (I)	CCLK	CCLK	CCLK	CCLK	60	74	A11	P1	R2	CCLK (I)
		$\overline{\text{WS}}$ (I)	A0	A0	61	75	B10	M2	P2	I/O
		CS2 (I)	A1	A1	62	76	B9	N1	M3	I/O
			A2	A2	63	77	A10	L2	P1	I/O
			A3	A3	64	78	A9	L1	N1	I/O
			A15	A15	65	81	B6	K1	M1	I/O
			A4	A4	66	82	B7	J2	L2	I/O
			A14	A14	67	83	A7	H1	K2	I/O
			A5	A5	68	84	C7	H2	K1	I/O
GND	GND	GND	GND	GND	1	1	C6	H3	J3	GND
			A13	A13	2	2	A6	G2	H2	I/O
			A6	A6	3	3	A5	G1	H1	I/O
			A12	A12	4	4	B5	F2	F2	I/O
			A7	A7	5	5	C5	E1	E1	I/O
			A11	A11	6	8	A3	D1	D1	I/O
			A8	A8	7	9	A2	D2	C1	I/O
			A10	A10	8	10	B3	B1	E3	I/O
			A9	A9	9	11	A1	C2	C2	I/O

▨ REPRESENTS A 50- TO 100-k Ω PULL-UP DURING CONFIGURATION
INIT IS AN OPEN DRAIN DURING CONFIGURATION

Note: Pin assignments of "PGA Footprint" PLCC sockets and PGA packages are not electrically identical.

Table 2b. Am3000 Family 68-Pin PLCC Pinouts

PGA Pin Number	Am3020	PGA Pin Number	Am3020	PGA Pin Number	Am3020
10	$\overline{PWRDN}$	32	I/O	55	D2–I/O
11	TCKLIN–I/O	33	I/O	N/C	UNBONDED IOB
12	I/O	34	$\overline{INIT}$–I/O	56	D1–I/O
13	I/O	35	GND	57	RDY/BUSY–RCLK–I/O
N/C	UNBONDED IOB	36	I/O	58	D0–DIN–I/O
14	I/O	37	I/O	59	DOUT–I/O
15	I/O	38	I/O	60	CCLK
16	I/O	39	I/O	61	A0–WS–I/O
17	I/O	40	I/O	62	A1–CS2–I/O
18	V_{CC}	41	I/O	63	A2–I/O
19	I/O	42	I/O	64	A3–I/O
N/C	UNBONDED IOB	43	XTL2(IN)–I/O	65	A15–I/O
20	I/O	44	$\overline{RESET}$	66	A4–I/O
21	I/O	45	DONE/$\overline{PROG}$	67	A14–I/O
22	I/O	46	D7–I/O	68	A5–I/O
N/C	UNBONDED IOB	47	XTL1(OUT)–BCLKIN–I/O	1	GND
23	I/O	48	D6–I/O	2	A13–I/O
24	I/O	N/C	UNBONDED IOB	3	A6–I/O
25	M1–$\overline{RDATA}$	49	D5–I/O	4	A12–I/O
26	M0–RTRIG	50	$\overline{CS0}$–I/O	5	A7–I/O
27	M2–I/O	51	D4–I/O	6	A11–I/O
28	HDC–I/O	N/C	UNBONDED IOB	7	A8–I/O
29	I/O	52	V_{CC}	8	A10–I/O
30	$\overline{LDC}$–I/O	53	D3–I/O	9	A9–I/O
31	I/O	54	$\overline{CS1}$–I/O		

The default configuration of IOBs is input with pull-up. This can be used to prevent an undefined pad level for unbonded or unused IOBs.

Table 2c. Am3000 Family 84-Pin PLCC and PGA Pinouts

PLCC Pin Number	PGA Pin Number	Am3020	Am3030	PLCC Pin Number	PGA Pin Number	Am3020	Am3030
12	B2	$\overline{\text{PWRDN}}$	$\overline{\text{PWRDN}}$	54	K10	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$
13	C2	TCLKIN–I/O	TCLKIN–I/O	55	J10	DONE–$\overline{\text{PROG}}$	DONE–$\overline{\text{PROG}}$
14	B1	N/C	I/O	56	K11	D7–I/O	D7–I/O
15	C1	I/O	I/O	57	J11	XTL1(OUT)–BCLKIN–I/O	XTL1(OUT)–BCLKIN–I/O
16	D2	I/O	I/O	58	H10	D6–I/O	D6–I/O
17	D1	I/O	I/O	59	H11	I/O	I/O
18	E3	I/O	I/O	60	F10	D5–I/O	D5–I/O
19	E2	I/O	I/O	61	G10	$\overline{\text{CS0}}$–I/O	$\overline{\text{CS0}}$–I/O
20	E1	I/O	I/O	62	G11	D4–I/O	D4–I/O
21	F2	I/O	I/O	63	G9	I/O	I/O
22	F3	V_{CC}	V_{CC}	64	F9	V_{CC}	V_{CC}
23	G3	I/O	I/O	65	F11	D3–I/O	D3–I/O
24	G1	I/O	I/O	66	E11	$\overline{\text{CS1}}$–I/O	$\overline{\text{CS1}}$–I/O
25	G2	I/O	I/O	67	E10	D2–I/O	D2–I/O
26	F1	I/O	I/O	68	E9	I/O	I/O
27	H1	I/O	I/O	69	D11	N/C	I/O
28	H2	I/O	I/O	70	D10	D1–I/O	D1–I/O
29	J1	I/O	I/O	71	C11	RDY/$\overline{\text{BUSY}}$–$\overline{\text{RCLK}}$–I/O	RDY/$\overline{\text{BUSY}}$–$\overline{\text{RCLK}}$–I/O
30	K1	I/O	I/O	72	B11	D0–DIN–I/O	D0–DIN–I/O
31	J2	M1–$\overline{\text{RDATA}}$	M1–$\overline{\text{RDATA}}$	73	C10	DOUT–I/O	DOUT–I/O
32	L1	M0–$\overline{\text{RTRIG}}$	M0–$\overline{\text{RTRIG}}$	74	A11	CCLK	CCLK
33	K2	M2–I/O	M2–I/O	75	B10	A0–$\overline{\text{WS}}$–I/O	A0–$\overline{\text{WS}}$–I/O
34	K3	HDC–I/O	HDC–I/O	76	B9	A1–CS2–I/O	A1–CS2–I/O
35	L2	I/O	I/O	77	A10	A2–I/O	A2–I/O
36	L3	$\overline{\text{LDC}}$–I/O	$\overline{\text{LDC}}$–I/O	78	A9	A3–I/O	A3–I/O
37	K4	I/O	I/O	79	B8	N/C	I/O
38	L4	N/C	I/O	80	A8	N/C	I/O
39	J5	I/O	I/O	81	B6	A15–I/O	A15–I/O
40	K5	I/O	I/O	82	B7	A4–I/O	A4–I/O
41	L5	N/C	I/O	83	A7	A14–I/O	A14–I/O
42	K6	$\overline{\text{INIT}}$–I/O	$\overline{\text{INIT}}$–I/O	84	C7	A5–I/O	A5–I/O
43	J6	GND	GND	1	C6	GND	GND
44	J7	I/O	I/O	2	A6	A13–I/O	A13–I/O
45	L7	I/O	I/O	3	A5	A6–I/O	A6–I/O
46	K7	I/O	I/O	4	B5	A12–I/O	A12–I/O
47	L6	I/O	I/O	5	C5	A7–I/O	A7–I/O
48	L8	I/O	I/O	6	A4	N/C	I/O
49	K8	I/O	I/O	7	B4	N/C	I/O
50	L9	N/C	I/O	8	A3	A11–I/O	A11–I/O
51	L10	N/C	I/O	9	A2	A8–I/O	A8–I/O
52	K9	I/O	I/O	10	B3	A10–I/O	A10–I/O
53	L11	XTL2(IN)–I/O	XTL2–I/O	11	A1	A9–I/O	A9–I/O

The default configuration of IOBs is input with pull-up. This can be used to prevent an undefined pad level for unbonded or unused IOBs.

Table 2d. Am3000 Family 132-Pin PGA Pinouts

PGA Pin Number	Am3042	Am3064	PGA Pin Number	Am3042	Am3064	PGA Pin Number	Am3042	Am3064
C4	GND	GND	F12	I/O	I/O	N6	N/C	I/O
A1	$\overline{\text{PWRDN}}$	$\overline{\text{PWRDN}}$	E14	I/O	I/O	P5	N/C	I/O
C3	I/O	I/O	F13	I/O	I/O	M6	D2–I/O	D2–I/O
B2	I/O	I/O	F14	I/O	I/O	N5	I/O	I/O
B3	I/O	I/O	G13	I/O	I/O	P4	I/O	I/O
A2	N/C	I/O	G14	$\overline{\text{INIT}}$–I/O	$\overline{\text{INIT}}$–I/O	P3	I/O	I/O
B4	I/O	I/O	G12	V_{CC}	V_{CC}	M5	D1–I/O	D1–I/O
C5	I/O	I/O	H12	GND	GND	N4	$\overline{\text{RCLK}}$–$\overline{\text{BUSY}}$/RDY–I/O	$\overline{\text{RCLK}}$–$\overline{\text{BUSY}}$/RDY–I/O
A3	N/C	I/O	H14	I/O	I/O	P2	I/O	I/O
A4	I/O	I/O	H13	I/O	I/O	N3	I/O	I/O
B5	I/O	I/O	J14	I/O	I/O	N2	D0–DIN–I/O	D0–DIN–I/O
C6	I/O	I/O	J13	I/O	I/O	M3	DOUT–I/O	DOUT–I/O
A5	I/O	I/O	K14	I/O	I/O	P1	CCLK	CCLK
B6	I/O	I/O	J12	I/O	I/O	M4	V_{CC}	V_{CC}
A6	I/O	I/O	K13	I/O	I/O	L3	GND	GND
B7	I/O	I/O	L14	N/C	I/O	M2	A0–$\overline{\text{WS}}$–I/O	A0–$\overline{\text{WS}}$–I/O
C7	GND	GND	L13	I/O	I/O	N1	A1–CS2–I/O	A1–CS2–I/O
C8	V_{CC}	V_{CC}	K12	I/O	I/O	M1	I/O	I/O
A7	I/O	I/O	M14	I/O	I/O	K3	I/O	I/O
B8	I/O	I/O	N14	I/O	I/O	L2	A2–I/O	A2–I/O
A8	I/O	I/O	M13	XTL2–I/O	XTL2–I/O	L1	A3–I/O	A3–I/O
A9	I/O	I/O	L12	GND	GND	K2	I/O	I/O
B9	I/O	I/O	P14	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$	J3	I/O	I/O
C9	I/O	I/O	M11	V_{CC}	V_{CC}	K1	A15–I/O	A15–I/O
A10	I/O	I/O	N13	DONE–$\overline{\text{PG}}$	DONE–$\overline{\text{PG}}$	J2	A4–I/O	A4–I/O
B10	I/O	I/O	M12	D7–I/O	D7–I/O	J1	N/C	I/O
A11	N/C	I/O	P13	XTL1–I/O	XTL1–I/O	H1	A14–1/O	A14–1/O
C10	I/O	I/O	N12	I/O	I/O	H2	A5–I/O	A5–I/O
B11	I/O	I/O	P12	I/O	I/O	H3	GND	GND
A12	N/C	I/O	N11	D6–I/O	D6–I/O	G3	V_{CC}	V_{CC}
B12	I/O	I/O	M10	I/O	I/O	G2	A13–I/O	A13–I/O
A13	N/C	I/O	P11	N/C	I/O	G1	A6–I/O	A6–I/O
C12	I/O	I/O	N10	I/O	I/O	F1	N/C	I/O
B13	M1–$\overline{\text{RD}}$	M1–$\overline{\text{RD}}$	P10	I/O	I/O	F2	A12–I/O	A12–I/O
C11	GND	GND	M9	D5–I/O	D5–I/O	E1	A7–I/O	A7–I/O
A14	M0–RT	M0–RT	N9	$\overline{\text{CS0}}$–I/O	$\overline{\text{CS0}}$–I/O	F3	I/O	I/O
D12	V_{CC}	V_{CC}	P9	N/C	I/O	E2	I/O	I/O
C13	M2–I/O	M2–I/O	P8	N/C	I/O	D1	A11–I/O	A11–I/O
B14	HDC–I/O	HDC–I/O	N8	D4–I/O	D4–I/O	D2	A8–I/O	A8–I/O
C14	I/O	I/O	P7	I/O	I/O	E3	I/O	I/O
E12	I/O	I/O	M8	V_{CC}	V_{CC}	C1	I/O	I/O
D13	I/O	I/O	M7	GND	GND	B1	A10–I/O	A10–I/O
D14	$\overline{\text{LDC}}$–I/O	$\overline{\text{LDC}}$–I/O	N7	D3–I/O	D3–I/O	C2	A9–I/O	A9–I/O
E13	N/C	I/O	P6	$\overline{\text{CS1}}$–I/O	$\overline{\text{CS1}}$–I/O	D3	V_{CC}	V_{CC}

Table 2e. Am3000 Family 175-Pin PGA Pinouts

PGA Pin Number	Am3090	PGA Pin Number	Am3090	PGA Pin Number	Am3090	PGA Pin Number	Am3090
B2	$\overline{\text{PWRDN}}$	D13	I/O	R14	DONE–$\overline{\text{PROG}}$	R3	D0–DIN–I/O
D4	TCLKIN–I/O	B14	M1–$\overline{\text{RDATA}}$	N13	D7–I/O	N4	DOUT–I/O
B3	I/O	C14	GND	T14	XTL1(OUT)–BCLKIN–I/O	R2	CCLK
C4	I/O	B15	M0–$\overline{\text{RTRIG}}$	P13	I/O	P3	Vcc
B4	I/O	D14	Vcc	R13	I/O	N3	GND
A4	I/O	C15	M2–I/O	T13	I/O	P2	A0–$\overline{\text{WS}}$–I/O
D5	I/O	E14	HDC–I/O	N12	I/O	M3	A1–CS2–I/O
C5	I/O	B16	I/O	P12	D6–I/O	R1	I/O
B5	I/O	D15	I/O	R12	I/O	N2	I/O
A5	I/O	C16	I/O	T12	I/O	P1	A2–I/O
C6	I/O	D16	$\overline{\text{LDC}}$–I/O	P11	I/O	N1	A3–I/O
D6	I/O	F14	I/O	N11	I/O	L3	I/O
B6	I/O	E15	I/O	R11	I/O	M2	I/O
A6	I/O	E16	I/O	T11	D5–I/O	M1	A15–I/O
B7	I/O	F15	I/O	R10	$\overline{\text{CS0}}$–I/O	L2	A4–I/O
C7	I/O	F16	I/O	P10	I/O	L1	I/O
D7	I/O	G14	I/O	N10	I/O	K3	I/O
A7	I/O	G15	I/O	T10	I/O	K2	A14–I/O
A8	I/O	G16	I/O	T9	I/O	K1	A5–I/O
B8	I/O	H16	I/O	R9	D4–I/O	J1	I/O
C8	I/O	H15	$\overline{\text{INIT}}$–I/O	P9	I/O	J2	I/O
D8	GND	H14	Vcc	N9	Vcc	J3	GND
D9	Vcc	J14	GND	N8	GND	H3	Vcc
C9	I/O	J15	I/O	P8	D3–I/O	H2	A13–I/O
B9	I/O	J16	I/O	R8	$\overline{\text{CS1}}$–I/O	H1	A6–I/O
A9	I/O	K16	I/O	T8	I/O	G1	I/O
A10	I/O	K15	I/O	T7	I/O	G2	I/O
D10	I/O	K14	I/O	N7	I/O	G3	I/O
C10	I/O	L16	I/O	P7	I/O	F1	I/O
B10	I/O	L15	I/O	R7	D2–I/O	F2	A12–I/O
A11	I/O	M16	I/O	T6	I/O	E1	A7–I/O
B11	I/O	M15	I/O	R6	I/O	E2	I/O
D11	I/O	L14	I/O	N6	I/O	F3	I/O
C11	I/O	N16	I/O	P6	I/O	D1	A11–I/O
A12	I/O	P16	I/O	T5	I/O	C1	A8–I/O
B12	I/O	N15	I/O	R5	D1–I/O	D2	I/O
C12	I/O	R16	I/O	P5	RDY/$\overline{\text{BUSY}}$–$\overline{\text{RCLK}}$–I/O	B1	I/O
D12	I/O	M14	I/O	N5	I/O	E3	A10–I/O
A13	I/O	P15	XTL2(IN)–I/O	T4	I/O	C2	A9–I/O
B13	I/O	N14	GND	R4	I/O	D3	Vcc
C13	I/O	R15	$\overline{\text{RESET}}$	P4	I/O	C3	GND
A14	I/O	P14	Vcc				

The default configuration of IOBs is input with pull-up. This can be used to prevent an undefined pad level for unbonded or unused IOBs. Pins A2, A3, 15, A16, T1, T2, T3, and T16 are not connected. Pin A1 does not exist.

ABSOLUTE MAXIMUM RATINGS

V_{CC}	Supply voltage relative to GND	–0.5 to 7.0 V
V_{IN}	Input voltage with respect to GND	–0.5 to V_{CC} +0.5 V
V_{TS}	Voltage applied to three-state output	–0.5 to V_{CC} +0.5 V
V_{STG}	Storage temperature (ambient)	–65 to 150°C
T_{SOL}	Maximum soldering temperature (10 sec @ 1/16 in.)	+260°C
T_J	Junction temperature	
	Plastic	+125°C
	Ceramic	+200°C

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure of the device to absolute maximum rating conditions for extended periods may affect device reliability.

OPERATING RANGES

V_{CC} Supply voltage relative to GND

Commercial (C) Devices		
	Ambient Temperature (T_A)	0 to +70°C
	Supply Voltage (V_{CC})	4.75 V to 5.25 V
Industrial		
	Case Temperature (T_C)	–40°C to +70°C
	Supply Voltage (V_{CC})	4.5 V to 5.5 V
Military		
	Case Temperature (T_C)	–55°C to +125°C
	Supply Voltage (V_{CC})	4.5 V to 5.5 V
V_{IHT}	High-level input voltage — TTL configuration	2.0 to V_{CC}
V_{ILT}	Low-level input voltage — TTL configuration	0 V to 0.8 V
V_{IHC}	High-level input voltage — CMOS configuration	0.7 V_{CC} to V_{CC}
V_{ILC}	Low-level input voltage — CMOS configuration	0 V to 0.2 V_{CC}
T_{IN}	Input signal transition time	250 ns

Operating ranges define those limits between which the functionality of the device is guaranteed.

* *Military products 100% tested at T_C = +25°C, +125°C, and –55°C.*

DC CHARACTERISTICS Over Operating Range Unless Otherwise Specified (Note 1)

Parameter Symbol	Parameter Description	Test Conditions		Min.	Max.	Unit
V_{OH}	High-level output voltage	I_{OH} = –4.0 mA V_{CC} min	Commercial	3.86		V
			Industrial	3.76		V
			Military	3.7		V
V_{OL}	Low-level output voltage	I_{OL} = 4.0 mA V_{CC} min	Commercial		0.32	V
			Industrial		0.37	V
			Military		0.4	V
I_{CCPD}	Power-down supply current	V_{CC} = 5.0 V @ 70°C				
	Am3020				500	µA
	Am3030				800	µA
	Am3042				1150	µA
	Am3064				1650	µA
	Am3090				2500	µA
I_{CCO}	Quiescent LCA supply current in addition to I_{CCPD} (Note 2)					
	CMOS chip thresholds		Com/Industrial		500	µA
			Military		1	mA
	TTL chip thresholds		Com/Industrial		10	mA
			Military		15	mA
I_{IL}	Leakage current Temperature		Com/Industrial	–10	+10	µA
			Military	–20	+15	µA
C_{IN}	Input capacitance all packages except PGA 175 (Note 3)	V_{CC} = 5.0 V, T_A = 25°C, f = 1.0 MHz				
	All pins except XTL1 and XTL2				10	pF
	XTL1 and XTL2				15	pF
	Input capacitance PGA 175 (Note 3)	V_{CC} = 5.0 V, T_A = 25°C, f = 1.0 MHz				
	All pins except XTL1 and XTL2				15	pF
	XTL1 and XTL2				20	pF
I_{RIN}	Pad pull-up (when selected)	V_{IN} = 0 V		0.02	0.17	mA
I_{RLL}	Horizontal long line pull-up (when selected) (Note 3)	logic LOW		0.4	3.4	mA

Notes:

1. For APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted.
2. With no output current loads, no active input or long line pull-up resistors, all package pins at V_{CC} or GND, and the LCA configured with a MAKEBITS "ie" option. See LCA power chart for additional activity dependent operating component.
3. These parameters are not 100% tested, but are evaluated at initial characterization and at any time the design is modified where capacitance may be affected.

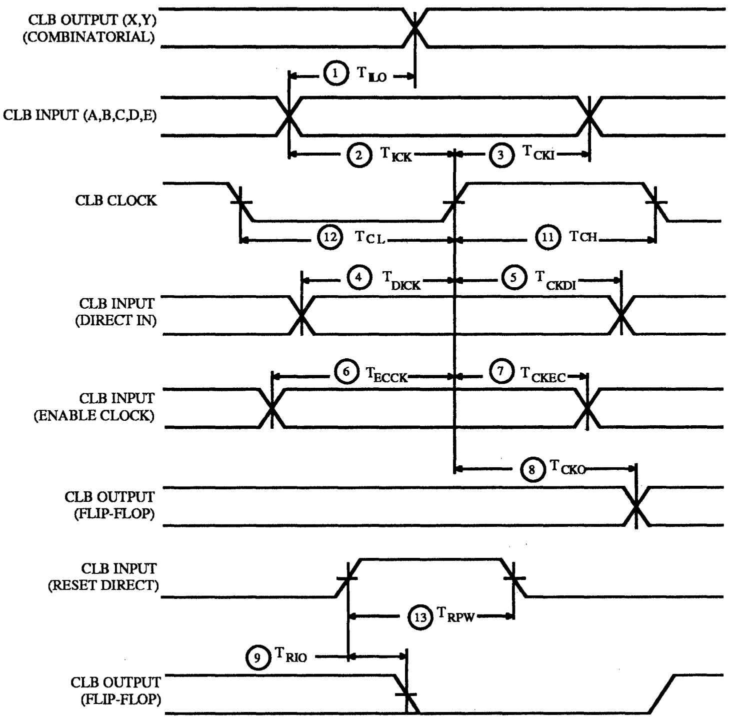
BENCHMARK PATTERNS*
APL Products

Part Number	Test	Symbol	Conditions $-55°C \le T_C \le +125°C$ $V_{CC} = 5.0\ V \pm 10\%$	Group A Subgroups	MIL Only** -50 Min.	MIL Only** -50 Max.	Unit
Am3020	T_{PID} + interconnect + 8 (T_{ILO}) + T_{OP}. Measured on 8 cols.	T_{B1}		9, 10, 11		136	ns
Am3030	T_{PID} + interconnect + 10 (T_{ILO}) + T_{OP}. Measured on 10 cols.	T_{B1}		9, 10, 11	TBD		ns
Am3042	T_{PID} + interconnect + 12 (T_{ILO}) + T_{OP}. Measured on 12 cols.	T_{B1}		9, 10, 11	TBD		ns
Am3064	T_{PID} + interconnect + 16 (T_{ILO}) + T_{OP}. Measured on 14 cols.	T_{B1}		9, 10, 11	TBD		ns
Am3090	T_{PID} + interconnect + 20 (T_{ILO}) + T_{OP}. Measured on 16 cols.	T_{B1}		9, 10, 11	TBD		ns

* Testing of the Applications Guidelines is modeled after testing specified by MIL-M-38510/605. Devices are first 100% functionally tested. Benchmark patterns are then used to determine our compliance to the Application Guidelines. Characterization data are taken at initial device qualification, prior to introduction of significant changes, and at least twice yearly to monitor correlation between patterns, device performance, XACT software timings, and the data sheet.

**70-MHz data not available at time of print.

CLB Switching Characteristic Guidelines



CLB Switching Characteristic Guidelines (Continued)

Testing of the switching characteristic guidelines is modeled after testing specified by MIL-M-38510/605. Devices are 100% functionally tested. Benchmark timing patterns are then used to determine correlation to the switching characteristic guideline values. Actual worst-case timing is provided by the XACT Timing calculator or Simulation modeling.

				MIL/COM'L		MIL/COM'L		COM'L		
			Speed Grade	–50		–70		–100		Unit
	Description	Symbol		Min.	Max.	Min.	Max.	Min.	Max.	
CLB Logic input	Combinational	1	T_{ILO}		14		9		7	ns
Reset direct	CLB output	9	T_{RIO}		15		10		7	ns
	Reset Direct width*	13	T_{RPW}	10		7		7		ns
	Master Reset pin to CLB out		T_{MRQ}		35		25		17	ns
CLB K Clock input	To CLB output	8	T_{CKO}		12		8		7	ns
	Additional for Q returning through F or G to CLB out		T_{QLO}		11		7		5	ns
	Logic-input setup	2	T_{ICK}	12		8		7		ns
	Logic-input hold	3	T_{CKI}	1		1		0		ns
	Data in setup	4	T_{DICK}	8		5		4		ns
	Data in hold (Note 1)	5	T_{CKDI}	6		4		2		ns
	Enable Clock setup	6	T_{ECCK}	10		7		5		ns
	Enable Clock hold	7	T_{CKEC}	1		1		0		ns
	Clock (HIGH)*	11	T_{CH}	9		7		5		ns
	Clock (LOW)*	12	T_{CL}	9		7		5		ns
Flop-flop Toggle rate	Q through F/G to flip-flop in*		F_{CLK}	50		70		100		MHz

Notes:

1. The CLB K to Q output delay (T_{CKO}, #8) of any CLB , plus the shortest possible interconnect delay, is always longer than the Data in hold time requirement (T_{CKDI}, #5) of any CLB on the same die.

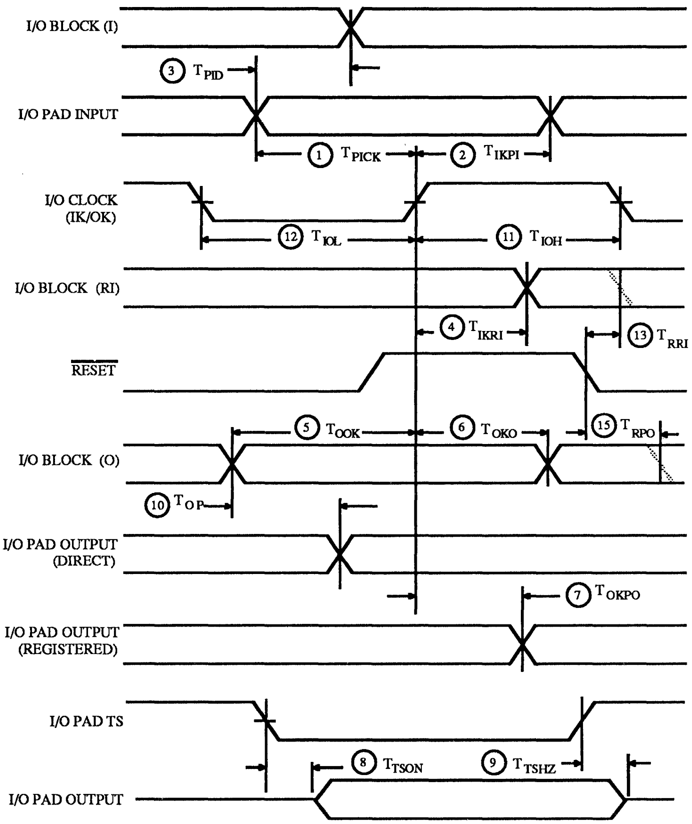
* These timing limits are based on calculations.
The speed of block inputs is a function of interconnect.

Buffer (Internal) Switching Characteristic Guidelines

		MIL/COM'L		MIL/COM'L		COM'L		
	Speed Grade	-50		-70		-100		Units
	Description	Min.	Max.	Min.	Max.	Min.	Max.	
Clock Buffer**	GCLK, ACLK		9		6		4	ns
TBUF**	Data to Output (Long line buffer)		8		5		4	ns
	Three-state to output							
	Single pull-up resistor		34		22		14	ns
	Pair of pull-up resistors		17		11		7	ns
Bi-directional	BIDI		6		4		3	ns

** Timing is based on the Am3020, for other devices see XACT timing calculator.

IOB Switching Characteristic Guidelines



IOB Switching Characteristic Guidelines (Continued)

Testing of the switching characteristic guidelines is modeled after testing specified by MIL-M-38510/605. Devices are 100% functionally tested. Benchmark timing patterns are then used to determine correlation to the switching characteristic guideline values. Actual worst-case timing is provided by the XACT Timing calculator or Simulation modeling.

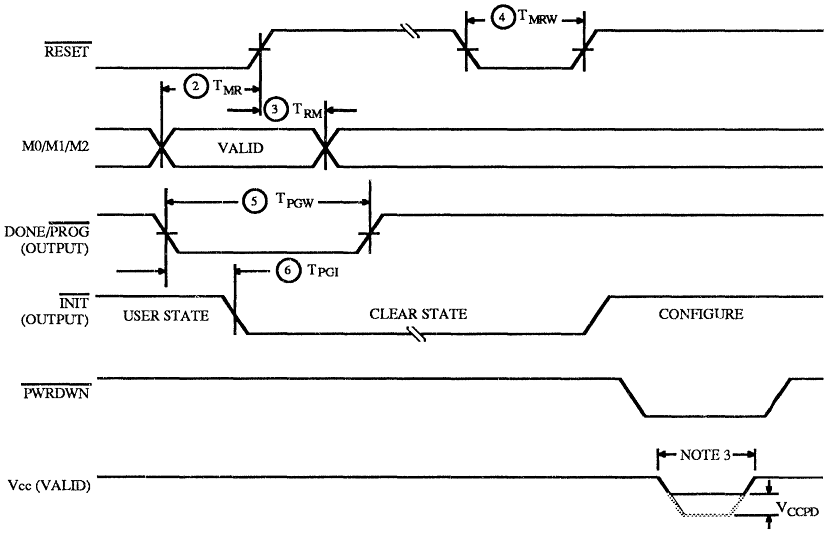
				MIL/COM'L		MIL/COM'L		COM'L		
			Speed Grade	–50		–70		–100		Unit
	Description		Symbol	Min.	Max.	Min.	Max.	Min.	Max.	
Pad (package pin)	To inputs TCLKIN, BCLKIN		T_{PIDC}		5		3		2	ns
	To inputs DIRECT IN	3	T_{PID}		10		7		4	ns
I/O Clock	To I/O RI input (FF)	4	T_{IKRI}		10		7		6	ns
	I/O pad-input setup	1	T_{PICK}	30		20		10		ns
	I/O pad-input hold	2	T_{IKPI}	0		0		0		ns
	To I/O pad (fast)	7	T_{OKPO}		18		13		10	ns
	I/O pad output setup	5	T_{OOK}	15		10		9		ns
	I/O pad output hold	6	T_{OKO}	0		0		0		ns
	Clock (HIGH)	11	T_{IOH}	9		7		5		ns
	Clock (LOW)	12	T_{IOL}	9		7		5		ns
Output	To pad (enabled fast)	10	T_{OPF}		14		10		6	ns
	To pad (enabled slow)	10	T_{OPS}		33		25		23	ns
Three-state	To pad begin hi-Z (fast)	9	T_{TSHZ}		12		8		8	ns
	To pad valid (fast)	8	T_{TSON}		20		14		12	ns
Master Reset	To input RI	13	T_{RRI}		45		30		20	ns
(Package Pin)	To output (FF)	15	T_{RPO}		55		37		28	ns

Notes:

1. Timing is measured at pin threshold, with 50 pF external capacitance loads (incl. test fixture). Typical fast mode output rise/ fall times are 2 ns and will increase approximately 2%/ pF. A maximum total external capacitive load for simultaneous fast mode switching in the same direction is 500 pF per power/ground pin pair.

2. Voltage levels of unused (bonded and unbonded) pads must be valid logic levels. Each can be configured with the internal pull-up resistor or alternatively configured as a driven output or driven from and external source,

General LCA Switching Characteristics



				COM'L		MIL		Unit
		Speed Grade		–50, –70, –100		–50, –70		
	Description	Symbol		Min.	Max.	Min.	Max.	
$\overline{\text{RESET}}$ (2) *	M2, M1, M0 setup	2	T_{MR}	1		1		µs
	M2, M1, M0 hold	3	T_{RM}	1		1		µs
	Width (low) Abort	4	T_{MRW}	6		6		µs
DONE/$\overline{\text{PROG}}$*	Program width (LOW)	5	T_{PGW}	6		6		µs
	Start $\overline{\text{INIT}}$	6	T_{PGI}		7		7	µs
$\overline{\text{PWRDWN}}$ (3) †	Power-Down V_{CC}		V_{CCPD}	2.0		3.5		V

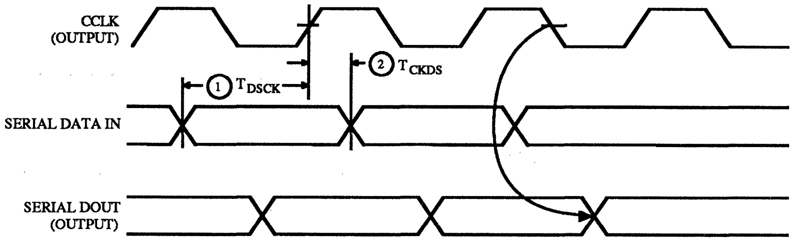
Notes:

1. V_{CC} must rise from 2.0 V for Commercial and 3.5 V for Military to V_{CC} minimum in less than 10 ms for master modes.
2. $\overline{\text{RESET}}$ timing relative to valid mode lines (M0, M1, M2) is relevant when $\overline{\text{RESET}}$ is used to delay configuration.
3. $\overline{\text{PWRDWN}}$ transitions must occur during operational V_{CC} levels.

* For APL Products, Group A, Subgroups 9, 10, 11 are tested under setup conditions.

† For APL Products, Group A, Subgroups 1, 2, 3 are tested under setup conditions.

Master Serial Mode Switching Characteristics Guidelines



				COM'L		MIL		
		Speed Grade		–50, –70, –100		–50, –70		Unit
	Description	Symbol		Min.	Max.	Min.	Max.	
CCLK (Note 3)	Data in setup	1	T_{DSCK}	60		60		ns
	Data in hold	2	T_{CKDS}	0		0		ns

Notes:

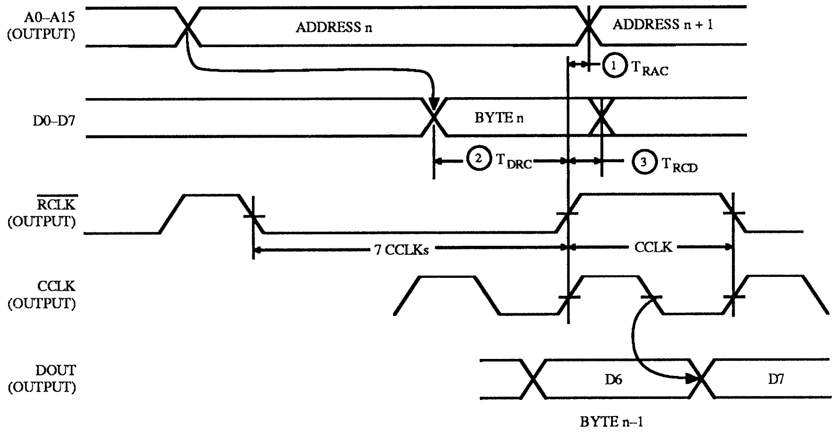
1. At power-up, V_{CC} must rise from 2.0 V for Commercial and 3.5 V for Military to V_{CC} minimum in less than 10 ms, otherwise delay configuration using $\overline{\text{RESET}}$ until V_{CC} is valid.

2. Configuration can be controlled by holding $\overline{\text{RESET}}$ LOW with or until after the $\overline{\text{INIT}}$ of all daisy-chain slave mode devices is HIGH.

3. Master serial mode timing is based on slave mode testing.

Master Parallel Mode Programming Switching Characteristics Guidelines

Testing of the switching characteristic guidelines is modeled after testing specified by MIL-M-38510/605. Devices are 100% functionally tested. Benchmark timing patterns are used to provide correlation to the switching characteristic guideline values. Actual worst-cast timing is provided by the XACT Timing calculator or Simulation modeling.

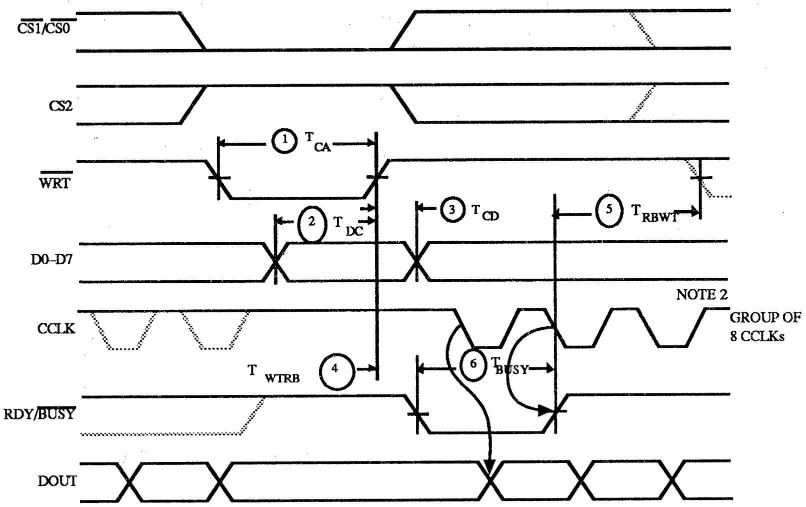


				COM'L		MIL		
		Speed Grade		-50, -70, -100		-50, -70		Unit
	Description	Symbol		Min.	Max.	Min.	Max.	
RCLK	To address valid	1	T_{RAC}	0	200	0	200	ns
	To data setup	2	T_{DRC}	60		60		ns
	To data hold	3	T_{RCD}	0		0		ns
	RCLK HIGH		T_{RCH}	600		600		ns
	RCLK LOW		T_{RCL}	4.0		4.0		µs

Notes:

1. At power-up, V_{CC} must rise from 2.0 V for Commercial and 3.5 V for Military to V_{CC} minimum in less than 10 ms, otherwise delay configuration using $\overline{\text{RESET}}$ until V_{CC} is valid.
2. Configuration can be controlled by holding $\overline{\text{RESET}}$ LOW with or until after the $\overline{\text{INIT}}$ of all daisy-chain slave mode devices is HIGH.

Peripheral Mode Programming Switching Characteristics



				COM'L		MIL		Unit
		Speed Grade		–50, –70, –100		–50, –70		
	Description	Symbol		Min.	Max.	Min.	Max.	
Write*	$\overline{\text{WRT}}$ Low	1	T_{CA}	0.5		0.5		µs
	DIN setup	2	T_{DC}	60		60		ns
	DIN hold	3	T_{CD}	0		0		ns
	Ready/$\overline{\text{Busy}}$	4	T_{WTRE}		60		60	ns
RDY*	WRT Active	5	T_{RBWT}	0		0		ns
	Busy	6	T_{BUSY}	4	9	4	9	CCLK

Notes:

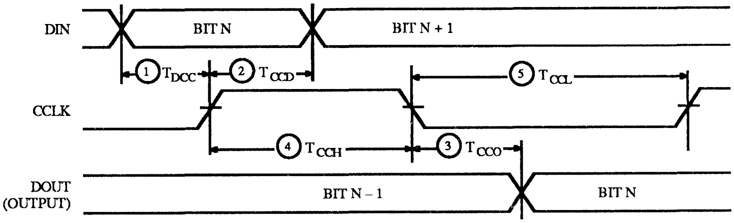
1. Configuration must be delayed until the $\overline{\text{INIT}}$ of all LCAs is HIGH.

2. Time from end of WRT to CCLK cycle for the new byte of data depends on completion of previous byte processing and the phase of the internal generator for CCLK.

3. CCLK and DOUT timing is tested in slave mode.

* For APL Products, Group A, Subgroups 9, 10, 11 are tested under setup conditions.

Slave Mode Programming Switching Characteristics

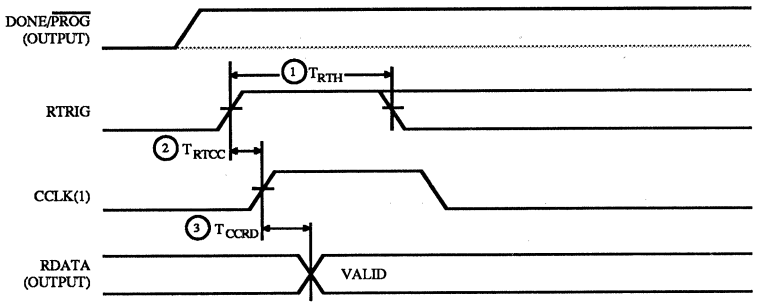


				COM'L		MIL		Unit
		Speed Grade		–50, –70, –100		–50, –70		
	Description	Symbol		Min.	Max.	Min.	Max.	
CCLK*	To DOUT	3	T_{CCO}		100		100	ns
	DIN setup	1	T_{DCC}	60		60		ns
	DIN hold	2	T_{CCD}	0		0		ns
	High time	4	T_{CCH}	0.5		0.5		µs
	Low time	5	T_{CCL}	0.5	5.0	0.5	5.0	µs
	Frequency		F_{CC}		1		1	MHz

Notes:

1. Configuration must be delayed until the $\overline{\text{INIT}}$ of all LCAs is HIGH.
   * For APL Products, Group A, Subgroups 9, 10, 11 are tested under setup conditions.

Program Readback Switching Characteristics



			Speed Grade	COM'L		MIL		Unit
				-50, -70, -100		-50, -70		
	Description	Symbol		Min.	Max.	Min.	Max.	
RTRIG*	RTRIG HIGH	1	T_{RTH}	250		250		ns
CCLK*	RTRIG setup	2	T_{RTCC}	10		200		µs
	$\overline{\text{RDATA}}$ delay	3	T_{CCRD}		100		100	ns

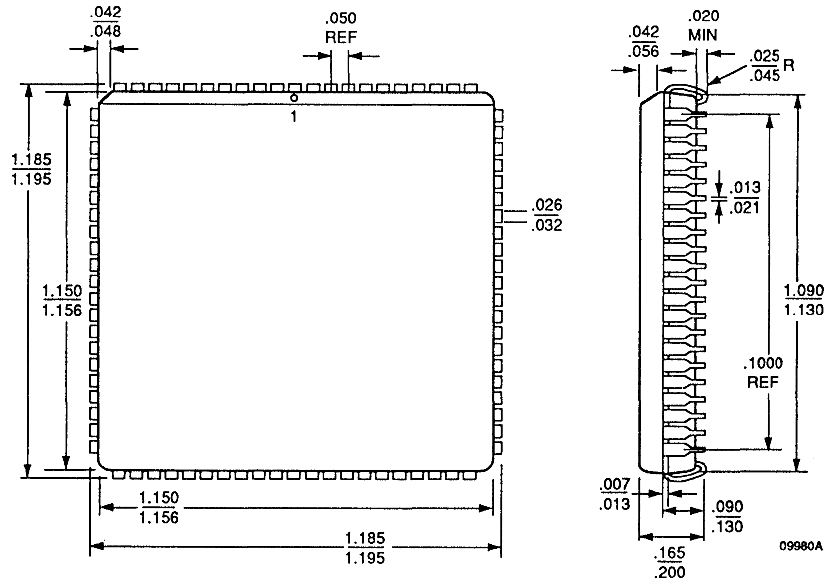
Notes:

1. CCLK and DOUT timing are the same as for slave mode.
2. RETRIG (M0 positive transition) shall not be done until after one clock following active I/O pins.
3. Readback should not be initiated until configuration is complete.

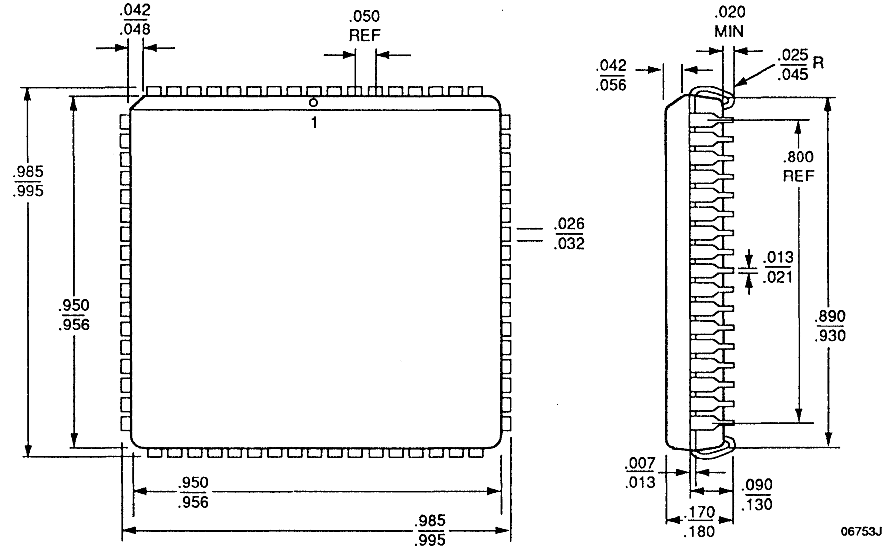
* For APL Products, Group A, Subgroups 9, 10, 11 are tested under setup conditions.

PHYSICAL DIMENSIONS*

84J Plastic Leaded Chip Carrier (PL 084)

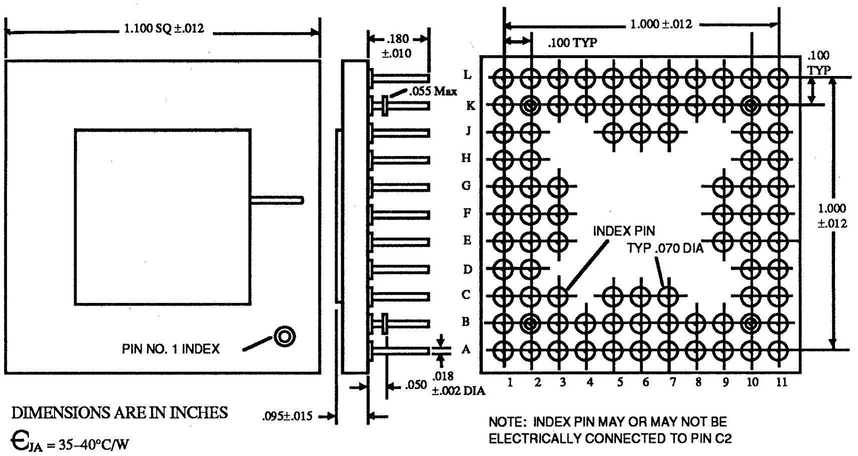


68J Plastic Leaded Chip Carrier (PL 068)



* For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.
** Package Under Development

PHYSICAL DIMENSIONS* (Continued)
84-Pin Ceramic Pin Grid Array**

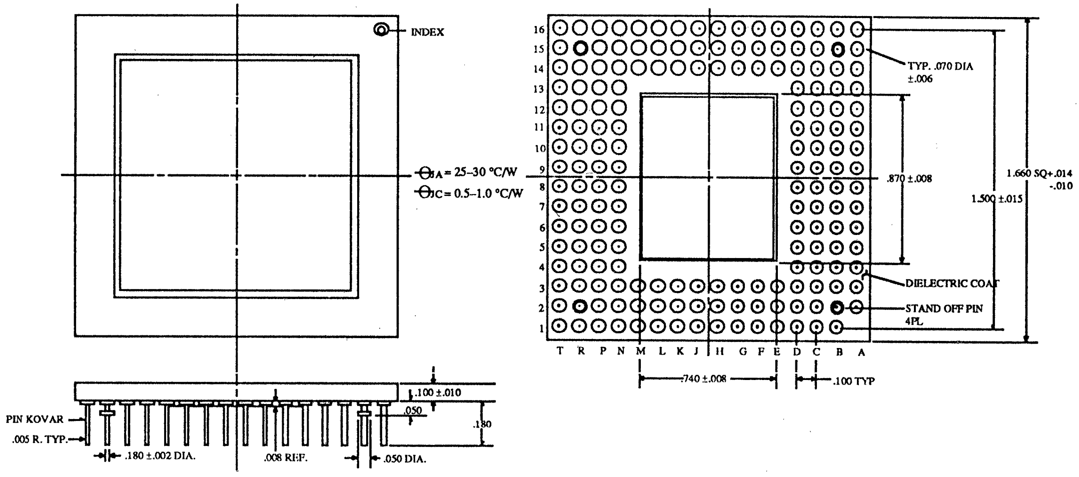


DIMENSIONS ARE IN INCHES

θ_{JA} = 35–40°C/W

NOTE: INDEX PIN MAY OR MAY NOT BE ELECTRICALLY CONNECTED TO PIN C2

175-Pin Ceramic Pin Grid Array**



Programmable Gate Array Software from AMD

Programmable Gate Arrays (PGAs) from AMD offer you higher levels of system integration with the advantages of user-programmability. AMD also provides a complete range of software packages to support PGA designs.

PGAs incorporate flexible blocks of configurable logic in a matrix fashion interfaced with a network of programmable interconnections. This network of logic blocks is surrounded by a ring of programmable I/O blocks. This unique architecture offers the capability to implement logic designs efficiently and effectively. On-chip logic allows configuration data to be loaded automatically at power-up, or the device can be reconfigured on-the-fly.

Designing with PGAs is done quickly and easily with modular software packages that run on the IBM-PC. The software allows you to enter a design, automatically place and route it, and verify function and timing, all within a matter of days. Once you are satisfied with the design, the appropriate configuration data can be created to personalize the device. Configuration data can reside in an EEPROM, EPROM, or ROM on the circuit board or on a floppy or hard disk.

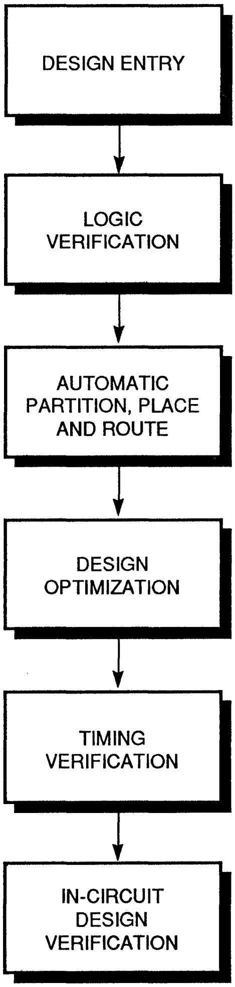
The discussion below outlines the tasks necessary to implement a PGA design. The complete range of software tools available from AMD allows you to accomplish gate array designs to 9000 equivalent gates at your desk, reducing design time, cost, and risk.

PGA Design Cycle

Designing with PGAs is a simple process, and AMD design tools are available for every phase in the design cycle. The diagram to the right shows the six steps in designing with PGAs.

Design entry can be accomplished by using the OrCAD/SDT™ III Schematic Design Tools, available as part of the PGA Bundled Development System (AmPGA151). For users with other schematic capture systems already in place, AMD also offers PGA interfaces to the Mentor™ and Daisy™ workstations and to the FutureNet™ Design package. In addition to the interfaces available from AMD, Valid™ workstation interfaces are available from Valid Logic Systems. In all cases, you use the PGA symbol library that is part of your schematic capture interface package.

Logic verification is an optional step that can be done at this point in the design cycle through the use of a logic simulator. The PGA Development System with Simulation (AmPGA251) includes the OrCAD/VST™ simulation tool. PC-SILOS™ (AmPGA022) is also available from AMD for simulation. Simulation capabilities are also available from supported CAE systems.



PGA Design Flow

Automatic partitioning, place and route is performed by the Automatic Design Implementation (ADI) software, included in each of the bundled development systems. Unused and redundant logic is eliminated and the design is partitioned into PGA resources (logic and I/O blocks). Automatic reduction and partitioning allows designers to immediately determine the PGA size required, during design definition and entry. During the automatic place and route phase of design, users can define timing constraints through critical nets and automatically place and route the total design.

Design optimization can be done either by returning to your schematic capture software or through the use of the PGA Design Editor. With the design editor, you can perform such tasks as moving blocks, rerouting nets and adding or deleting logic. A timing calculator, part of the design editor, permits point-to-point timing determinations for critical path analysis.

Timing verification is an optional step that can be done once the design has been placed and routed to your satisfaction.

In-circuit design verification is done in one of three ways. An in-circuit emulator is available to simulate the circuit (AmPGA028). In addition, during design debug, designers can save time by using the download cable to transfer the configuration program from the PC directly into a PGA under development. Once the design is complete, a PROM file is created to be loaded into a PROM via a PROM programmer.

You iterate these steps until your design is correct. You can quickly correct design errors by making changes in your original design and repeating the subsequent steps. Even if you detect an error during in-circuit design verification, you can make corrections quickly and easily.

PGA Bundled Development System with Simulation (AmPGA251)

The PGA Bundled Development System offers complete capabilities for the implementation of a Programmable Gate Array design. Packaged together are the OrCAD/SDT III Schematic Design Tools, OrCAD/VST Verification and Simulation Tools, Automatic Design Implementation, and the PGA Development System.

Each part of a bundled PGA system is designed to perform one of the PGA design tasks described above. Below is a summary of the features of each of the components of AMD's bundled PGA software.

OrCAD Schematic Design Tools

OrCAD/SDT III is a complete schematic package, designed to place Computer Aided Engineering power at the desk of every engineer. Easy to use menu driven commands make it possible to create, edit, save, and print design schematics. The graphical editing capabilities allow single objects or groups of objects to be easily moved, replicated, or deleted. OrCAD/SDT III can store over 100 individual macros, each executed with a single keystroke.

Designs can be entered hierarchically to manage the complexity of large designs. Designers can flag critical timing paths to ensure that critical signals are routed with minimum delay. A wide range of graphics boards are supported, with an extensive selection of printers and plotters for output.

In addition to the OrCAD/SDT III software, all bundled systems contain the interface necessary to design for PGAs, and the PGA macro library of almost 300 TTL and SSI/MSI standard family equivalents.

OrCAD Verification and Simulation Tools

OrCAD/VST is a full-featured, 12-state, event-driven logic simulator, capable of handling designs of 14,000 gates at an evaluation speed of 10,000 events per second, and can simulate more than 2 billion time units. The package integrates a stimulus generator, design linker, and design compiler into a simulation environment. A netlist is all that is required to get started. The link and compile steps are automatically performed, and the stimulus is defined within an integrated pop-up editor, enabling you to initialize signals, generate any kind of clock signal, and perform test vector definition.

Definition of signals that are to be traced or displayed is performed by the Trace Editor. Displays can be defined as signals or buses. Buses can be displayed as binary, octal, decimal or hexadecimal. OrCAD/VST's logic analyzer style format simplifies data analysis. Signals and bus values are easily viewed, displayed as a window into the trace buffer. Four zoom levels enable you to magnify the traces. Up to three markers may be placed on the screen to measure time intervals quickly.

In addition to the OrCAD/VST software, all bundled systems contain the interface necessary to use Simulation and Verification with PGAs, and the PGA macro library of almost 300 TTL and SSI/MSI standard family equivalents.

Automatic Design Implementation

The Automatic Design Implementation package enhances the productivity of designers using PGAs by reducing design placement and routing time, and at the same time, maintaining flexibility. Designs that are developed incrementally can take advantage of automatic placement and routing by locking partial PGA layouts in place while automatically placing and routing design additions.

The automatic placement and routing program is extremely flexible. Through placement directives, the user can control the placement process to achieve the best arrangement for a particular design. Routing resources can be specified to minimize clock skews and signal delays for critical paths. The result is faster product development.

PGA Design System

The PGA Development System provides users with a complete design and development system for specification and implementation of designs using PGAs. Functional definition of configurable logic blocks (CLBs), input/output blocks (IOBs), and interconnections is performed with a menu-driven interactive graphics editor. Functions are specified by CLB and IOB definitions plus their interconnections. The macro library and user-defined macros enable the user to easily implement complex functions. The check for logic connectivity and design rule violation is easily performed. All unused internal nodes are automatically configured to minimize power dissipation.

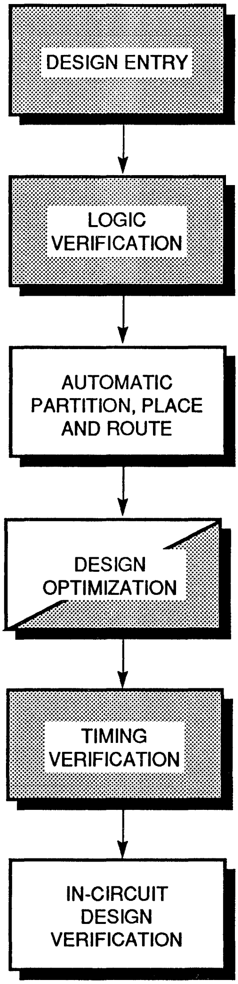
Interactive point-to-point timing delay calculation is provided for timing analysis and critical path determination. This ability enables the user to quickly identify and correct timing problems while the design is in progress.

A bundled development system without simulation (AmPGA151) is also available. Each piece of a bundled system may also be purchased individually. In addition, for those users who already have OrCAD software, a package containing only the PGA interface and library provides the necessary components to produce PGA designs using OrCAD tools (AmPGA035 and AmPGA045).

Workstation Support

With the AmPGA151 and AmPGA251 bundled development packages, AMD provides a complete, PC-based environment in which to design for PGAs. However, support is also available for a variety of CAE workstation products. In using one of these workstations, you move from the workstation environment to the PC environment and back, depending on the task. Below, in the diagram of the PGA design tasks, the shaded boxes represent tasks performed on the workstation and the white boxes denote PC-based tasks.

Currently, AMD supports workstations from Daisy/Cadnetix, Inc. and Mentor Graphics Corporation. A package is available from AMD for each of these systems, containing the PGA interface and library necessary to design for PGAs. In addition, the PGA interface and library for Valid EDA Systems is available from Valid Logic Systems.



PGA Design Flow

Daisy Schematic Capture and Simulation Interfaces (AmPGA133)

The Daisy interfaces from AMD allow the Daisy workstation user to enter a PGA design. It includes the following features:

- Full support of timing verification through back-annotation to the CLB or gate level
- Macro library of over 100 TTL and standard logic family equivalents
- User control of flagging critical paths
- Use of familiar Daisy design entry methodology
- Output compatibility with PGA Development Software
- Hardware, software support

Hardware platforms:

- IBM PC/AT
- Daisy

Software:

- DNIX Operating System
- ACE™/DED graphic schematic editors
- DLS™ Logic Simulator
- PGA library

Mentor Schematic Capture and Simulation Interfaces (AmPGA134)

The Mentor interfaces from AMD allow the Mentor workstation user to enter a PGA design. It includes the following features:

- Full support of timing verification through back-annotation to the CLB or gate level
- Use of familiar Mentor design entry methodology
- Macro library of over 100 TTL and standard logic family equivalents
- User control of flagging critical paths
- Output compatibility with PGA Development Software
- Hardware, software support

Hardware:

- All Apollo platforms

Software:

- Apollo Aegis Operating System
- IDEA Applications
- NETED™ Schematic Editor
- QuickSIM™ Logic Simulator

Valid Schematic Capture and Simulation Interfaces

The Valid/PGA interfaces from Valid Logic Systems allow the Valid EDA Systems user to enter a PGA design. It includes the following features:

- Full support of timing verification through back-annotation to the CLB level
- Use of familiar Valid design entry methodology
- Macro library of over 100 TTL and standard logic family equivalents
- Output compatibility with PGA Development Software
- Hardware, software support

Hardware:

- Sun
- Digital
- IBM PC/AT
- Valid

Software:

- ValidGED™ Graphics Editor
- ValidSIM™ Interactive Logic Simulator
- ValidTIME™ Timing Verifier

For more information about the Valid/PGA interface, contact

Valid Logic Systems
2820 Orchard Parkway
San Jose, California 95134
(408) 945-9400

Valid International
Valid House
39 Windsor Road
Slough Berkshire
SL1 2EE England
44 (75) 382 0101

Nihon Valid Logic Systems Co., Ltd.
Tokyo Building
2-16-8 Minami-Ikebukuro
Toshima-Ku, Tokyo 171, Japan
81 3 980 6421

Symbol Library

Each schematic interface package includes the LCA library, which must be used when entering designs to be implemented in an AMD Programmable Gate Array. The library contains both combinatorial logic primitives and macros. Primitives represent the lowest level of logic symbols. Macros are logic functions implemented by the use of primitives and/or other macros. The following table lists the functions that are available in the various interface libraries. An X denotes that the function is available in the LCA library for that interface.

		3000				2000			
Name	**Description**	**Design System**	**OrCAD**	**Daisy**	**Mentor**	**Design System**	**OrCAD**	**Daisy**	**Mentor**
BUFFERS:									
ACLK	Alternate clock buffer	X	X	X	X		X	X	X
GCLK	Global clock buffer	X	X	X	X		X	X	X
BUF1	Buffer			X	X			X	X
BUF	Buffer		X				X		
INV1	Inverter			X	X			X	X
INV	Inverter		X				X		
AND, NAND, NOR, OR, XOR, XNOR PRIMITIVES			X	X	X		X	X	X
GENERAL:									
GADD	Adder	X	X	X	X	X	X	X	X
GCOMP	Compare	X	X	X	X	X	X	X	X
GEQGT	Equal or greater-than comparator		X	X		X	X	X	X
GLTGT	Less-than and greater-than comparator	X	X	X	X			X	
GMAJ	4-to-1 majority gate	X	X	X		X	X	X	X
GOSC	Low frequency resistor-capacitor oscillator	X				X			
GXTL	Crystal oscillator		X	X	X	X	X	X	X
7483	4-bit binary full adder with fast carry		X				X		
7485	4-bit magnitude comparator		X				X		
74280	9-bit odd/even parity generator/checker		X				X		
74283	4-bit binary full adder with fast carry		X				X		
74518	8-bit identity comparator		X				X		
74521	8-bit identity comparator		X				X		
GPAR	Parity					X			
GXOR	Exclusive-OR					X			
GXOR2	Dual exclusive-OR					X			
INPUT/OUTPUT BUFFERS AND PADS:									
IBUF	Input buffer		X	X	X		X	X	X
OBUF	Output buffer		X	X	X		X	X	X
OBUFZ	Output buffer with 3-state control		X	X	X		X	X	X
INFF	IOB configured as a flip-flop		X	X	X		X	X	X
OSC	Oscillator		X	X	X		X	X	X
INLAT	IOB configured as an input latch		X	X	X				
BPAD	Bi-directional pad		X				X		
IPAD	Input pad		X				X		
UPAD	Unbonded pad		X				X		
OPAD	Output pad		X				X		
OUTFF	Output flip-flop		X	X	X				
OUTFFZ	Output flip-flop with 3-state control		X	X	X				
TBUF	Internal 3-state driver	X	X	X	X				
74125	3-state bus buffer		X						
74240	Inverted 3-state output buffer		X						
74241	Octal non-inverting 3-state output buffer		X						
74244	Octal non-inverting 3-state output buffer		X						
74245	Octal bidirectional transceiver		X						
74540	Octal buffer with 3-state outputs		X						
74541	Octal buffer with 3-state outputs		X						

Name	Description	3000				2000			
		Design System	OrCAD	Daisy	Mentor	Design System	OrCAD	Daisy	Mentor
LATCHES:									
LD	Data latch with load input	X	X	X	X	X	X	X	X
LDRD	Data latch with load input and reset direct	X	X	X	X	X	X	X	X
LDSD	Data latch with load input and set direct	X	X	X	X	X	X	X	X
LDSRD	Data latch with load input, reset and set direct		X			X	X	X	X
LDM	Data latch with 2-input data multiplexer		X	X		X	X	X	X
LDMRD	Data latch with 2-input data multiplexer and reset direct		X	X		X	X	X	X
LDMSD	Data latch with 2-input data multiplexer and set direct		X	X		X	X	X	X
DLAT	Data latch with set direct and reset direct		X				X	X	X
LRS	Data latch with set and reset	X	X	X	X			X	
7477	4-bit bi-stable latch		X				X		
74259	8-bit addressable latch		X				X		
74354	8-to-1 data selector/multiplexer/register		X						
74373	Octal latch		X						
FLIP-FLOPS:									
DFF	Positive edge-tiggered D flip-flop		X				X		
FD	D flip-flop with one data line	X	X	X	X	X	X	X	X
FDR	D flip-flop with reset	X	X	X	X	X	X	X	X
FDS	D flip-flop with set	X	X	X	X	X	X	X	X
FDRD	D flip-flop with reset direct	X	X	X	X	X	X	X	X
FDSD	D flip-flop with set direct					X	X	X	X
FDSRD	D flip-flop with set and reset direct					X	X	X	X
PDFF	D flip-flop with positive edge clock								X
NDFF	D flip-flop with negative edge clock								X
FDC	D flip-flop with clock enable	X	X	X	X	X	X	X	X
FDCR	D flip-flop with clock enable and reset	X	X	X	X	X	X	X	X
FDCS	D flip-flop with clock enable and set	X	X	X	X	X	X	X	X
FDM	D flip-flop with 2-input data multiplexer	X	X	X	X	X	X	X	X
FDMR	D flip-flop with 2-input data multiplexer and reset	X	X	X	X	X	X	X	X
FDMS	D flip-flop with 2-input data multiplexer and set	X	X	X	X	X	X	X	X
FDMRD	D flip-flop with 2-input data multiplexer and reset direct	X	X	X	X	X	X	X	X
FDMSD	D flip-flop with 2-input data multiplexer and set direct					X	X	X	X
FDCRD	D flip-flop with clock enable and reset direct	X	X	X	X				
DFF1	D flip-flop with set direct and reset direct			X	X			X	X
74174	Hex D flip-flop with clear		X				X		
74273	Octal D flip-flop with clear		X				X		
74374	Octal D flip-flop		X						
74377	Octal D flip-flop with enable		X				X		
74577	Octal D flip-flop with 3-state outputs		X						
FSR	Set and reset flip-flop with set dominant over reset	X	X	X	X	X	X	X	X
FRS	Set and reset flip-flop with reset dominance over set	X	X	X	X	X	X	X	X
FJK	J-K flip-flop	X	X	X	X	X	X	X	X
FJKS	J-K flip-flop with synchronous set	X	X	X	X	X	X	X	X
FJKRD	J-K flip-flop with reset direct	X	X	X	X	X	X	X	X
FJKSD	J-K flip-flop with set direct					X	X	X	X
FJKSRD	J-K flip-flop with set and reset direct					X	X	X	X
FT0	Self-toggle flip-flop	X	X	X	X	X	X	X	X
FT0R	Self-toggle flip-flop with reset	X	X	X	X	X	X	X	X
FT0RD	2-input toggle flip-flop with reset direct	X	X	X	X			X	
FT	Toggle flip-flop	X	X	X	X	X	X	X	X
FTP	Toggle flip-flop with parity enable	X	X	X	X	X	X	X	X
FTPRD	Toggle flip-flop with parity enable and reset direct	X	X	X	X	X	X	X	X

		3000				2000			
Name	**Description**	**Design System**	**OrCAD**	**Daisy**	**Mentor**	**Design System**	**OrCAD**	**Daisy**	**Mentor**
FTR	Toggle flip-flop with reset	X	X	X	X	X	X	X	X
FTRD	2-input toggle flip-flop	X	X	X	X		X	X	X
FTS	Toggle flip-flop with set	X	X	X	X	X	X	X	X
FT2	2-input toggle flip-flop		X	X		X	X	X	X
FT2R	2-input toggle flip-flop with reset		X	X		X	X	X	X
DECODERS/ENCODERS:									
D2-4	1-of-4 decoder	X	X	X	X	X	X	X	X
D2-4E	1-of-4 decoder with enable	X	X	X	X	X	X	X	X
74-139	1-of-4 single decoder with low output and enable	X	X	X	X	X	X	X	X
D3-8	1-of-8 decoder	X	X	X	X	X	X	X	X
D3-8E	1-of-8 decoder with enable	X	X	X	X	X	X	X	X
74-138	1-of-8 decoder with enables and low output	X	X	X	X	X	X	X	X
74-42	1-of-10 decoder with low output	X	X	X	X	X	X	X	X
7442	4-to-10-line decoder		X				X		
7448	BCD-to-7-segment decoder/driver		X				X		
74138	1-of-8 decoder/demultiplexer		X				X		
74139	1-of-4 decoder/demultiplexer		X				X		
74147	10-line-to-4-line priority encoder		X				X		
74148	8-line-to-3-line priority encoder		X				X		
74154	4-line-to-16-line decoder/demultiplexer		X				X		
MULTIPLEXERS:									
GMUX	2-to-1 mux	X	X	X	X	X	X	X	X
M3-1	3-to-1 multiplexer	X	X	X	X	X	X	X	X
M3-1E	3-to-1 multiplexer with enable	X	X	X	X	X	X	X	X
M4-1	4-to-1 multiplexer	X	X	X	X	X	X	X	X
M4-1E	4-to-1 multiplexer with enable	X	X	X	X	X	X	X	X
74-352	4-to-1 multiplexer with low output and enable	X	X	X	X	X	X	X	X
M4-2	4-to-2 multiplexer	X	X	X	X				X
M8-1	8-to-1 multiplexer	X	X	X	X	X	X	X	X
M8-1E	8-to-1 multiplexer with enable	X	X	X	X	X	X	X	X
74-151	8-to-1 multiplexer with enable and complementary outputs	X	X	X	X	X	X	X	X
74-152	8-to-1 multiplexer with low output	X	X	X	X	X	X	X	X
74151	8-input multiplexer with strobe		X				X		
74152	8-input multiplexer		X				X		
74153	Dual 4-line-to=1=line data selector/multiplexer		X				X		
74157	Quadruple 2-line-to-1-line data selector/multiplexer		X				X		
74158	Quadruple 2-line-to-2-line data selector/multiplexer		X				X		
74257	Quad data selector/multiplexer		X						
74258	Quad data selector/multiplexer		X						
74298	Quadruple 2-input multiplexer with storage		X				X		
74352	Dual 4-line-to-1-line data selector/multiplexor		X				X		
REGISTERS:									
RD4	4-bit data register with clock	X	X	X	X	X	X	X	X
RD8	8-bit data register with clock	X	X	XX	X	X	X	X	X
RD8CR	8-bit data register with clock enable and reset	X	X	X	X	X	X	X	X
RS4	4-bit shift register with clock	X	X	X	X	X	X	X	X
RS8	8-bit shift register	X	X	X	X	X	X	X	X
RS8CR	8-bit shift register with clock enable and reset	X	X	X	X	X	X	X	X
74-194	4-bit bidirectional shift register with clock, parity enable and reset direct	X	X	X	X	X	X	X	X

Name	Description	3000				2000			
		Design System	OrCAD	Daisy	Mentor	Design System	OrCAD	Daisy	Mentor
74-195	4-bit serial-to-parallel shift register with clock, parity enable and reset direct	X	X	X	X	X	X	X	X
RS8PR	8-bit shift register with clock, parity enable, and reset	X	X	X	X	X	X	X	X
RS8R	8-bit shift register with clock and reset	X	X	X	X	X	X	X	X
74-164	8-bit serial-to-parallet shift register with clock and reset direct	X	X	X	X	X	X	X	X
RD4RD	4-bit data register with clock and reset direct	X	X	X	X			X	
RD8RD	8-bit data register with clock and reset direct	X	X	X	X			X	
RS4C	4-bit shift register with clock enable	X	X	X	X			X	
RS4CR	4-bit shift register with clock enable and reset	X	X	X	X			X	
RS4CRD	4-bit shift register with clock enable and reset direct	X	X	X	X				
RS4RD	4-bit shift register with reset direct	X	X	X	X			X	
RS8C	8-bit shift register with clock enable	X	X	X	X			X	
RS8CRD	8-bit shift register with clock enable and reset direct	X	X	X	X				
RS8RD	8-bit shfit register with reset direct	X	X	X	X			X	
74164	8-bit parallel-out serial shift register		X				X		
74166	Parallel load 8-bit shift register		X				X		
74179	4-bit parallel-access shift register		X				X		
74194	4-bit bidirectional universal shift register		X				X		
74195	4-bit parallel-access shift register		X				X		
74198	8-bit bidirectional shift register		X				X		
74199	8-bit shift register with clock inhibit		X				X		
74278	4-bit cascadable priority register		X				X		
74299	8-bit universal shift register		X						
74323	8-to-1 universal shift/storage register		X						
74595	8-bit shift register with output register		X						
COUNTERS:									
Modulo 2:									
C2BCR	1-bit binary counter with clock and reset	X	X	X	X	X	X	X	X
C2BCRD	1-bit binary counter with clock and reset direct	X	X	X	X	X	X	X	X
C2BCP	1-bit binary counter with clock and parity enable	X	X	X	X			X	
C2BCPRD	1-bit binary counter with clock, parity enable and reset direct	X	X	X	X			X	
C2BP	1-bit binary counter with clock and parity enable	X	X	X	X	X	X	X	X
C2BR	1-bit binary counter with clock and reset	X	X	X	X	X	X	X	X
C2BRD	1-bit binary counter with reset direct	X	X	X	X	X	X	X	X
Modulo 4:									
C4BCP	2-bit binary counter with clock and parity enable	X	X	X	X	X	X	X	X
C4BCPRD	2-bit binary counter with clock enable and reset direct	X	X	X	X			X	
C4BCR	2-bit binary counter with clock enable and reset	X	X	X	X	X	X	X	X
C4BCRD	2-bit binary counter with clock enable and reset direct	X	X	X	X	X	X	X	X
C4JCR	2-bit Johnson counter with clock enable and reset		X	X		X	X	X	X
C4JX	2-bit shift expandable Johnson counter	X	X	X	X			X	
C4JXC	2-bit expandable Johnson counter with clock enable	X	X	X	X			X	
C4JXCR	2-bit expandable Johnson counter with clock enable and reset	X	X	X	X			X	
C4JXCRD	2-bit expandable Johnson counter with clock enable and reset direct	X	X	X	X				
C4JXRD	2-bit expandable Johnson counter with reset direct	X	X	X	X			X	
Modulo 6:									
C6JCR	3-bit Johnson counter with clock enable and reset	X	X	X	X	X	X	X	X

		3000				2000			
Name	Description	Design System	OrCAD	Daisy	Mentor	Design System	OrCAD	Daisy	Mentor
Modulo 8:									
C8BCP	3-bit binary counter with clock and parity enable	X	X	X	X	X	X	X	X
C8BCPRD	8-bit binary counter with clock, parity enable and reset direct	X	X	X	X			X	
C8BCR	3-bit binary counter with clock enable and reset	X	X	X	X	X	X	X	X
C8BCRD	3-bit binary counter with clock enable and reset direct	X	X	X	X	X	X	X	X
C8JCR	4-bit Johnson counter with clock enable and reset	X	X	X	X	X	X	X	X
Modulo 10:									
C10BCPRD	4-bit BCD counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
C10BCRD	4-bit BCD counter with clock enable and reset direct	X	X	X	X	X	X	X	X
74-160	4-bit BCD counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
74-162	4-bit binary counter with reset	X	X	X	X			X	
C10BPRD	4-bit BCD counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
C10JCR	5-bit Johnson counter with clock enable and reset	X	X	X	X	X	X	X	X
74160	Synchronous presettable 4-bit decade counter		X				X		
74162	Presettable decade counter with synchronous clear		X				X		
74168	Synchronous 4-bit up/down decade counter		X				X		
74390	4-bit decade counter with clear		X				X		
Modulo 12:									
C12JCR	6-bit Johnson counter with clock enable and reset	X	X	X	X	X	X	X	X
Modulo 16:									
C16BARD	4-bit binary ripple counter with clock and reset direct	X	X	X	X	X	X	X	X
C16BCRD	4-bit binary counter with clock enable and reset direct	X	X	X	X	X	X	X	X
C16BCPR	4-bit binary counter with clock, parity enable, and reset	X	X	X	X	X	X	X	X
C16BCPRD	4-bit binary counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
C16BCP	4-bit binary counter with clock, parity enable	X	X	X	X				
74-161	4-bit binary counter with clock and reset direct	X	X	X	X	X	X	X	X
74-163	4-bit binary counter with reset	X	X	X	X				
C16BPRD	4-bit binary up counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
C16BUDRD	4-bint binary up-down counter with clock, parity enable, and reset direct	X	X	X	X	X	X	X	X
C16JCR	8-bit Johnson counter with clock enable and reset	X	X	X	X	X	X	X	X
74161	Synchronous presettable 4-bit binary counter		X				X		
74163	Synchronous binary counter with synchronous clear		X				X		
74169	Synchronous 4-bit up/down binary counter		X				X		
74393	4-bit binary counter with clear		X				X		
74590	8-bit binary counter with output register		X						
Modulo 256:									
C256FCRD	8-bit modulo 256 feedback shift register with clock enable and reset direct	X	X	X	X	X	X	X	X
C256BCP	8-bit binary counter with clock, parity enable	X	X	X	X			X	
C256BCPR	8-bit binary counter with clock, parity enable, and reset direct	X	X	X	X			X	
C256BCR	8-bit binary counter with clock enable and reset	X	X	X	X			X	
C256BCRD	8-bit binary counter with clock enable and reset direct	X	X	X	X				

		3000				2000			
Name	Description	Design System	OrCAD	Daisy	Mentor	Design System	OrCAD	Daisy	Mentor
FLAGS:									
C	Critical signal flag		X				X		
L	Longline signal flag		X				X		
N	Non-critical signal flag		X				X		
X	Explicit signal flag		X				X		
K	K pin for clock		X				X		
G	G pin for clock		X				X		
I	Input (C) pin for clock		X				X		
CLB									
CLB	CLB primitive		X				X		
IOB PRIMITIVES:									
IOB	IOB primitive		X				X		
PADS:									
PIN	Input pad	X				X			
PINQ	Input pad with registered input	X				X			
PINR	Input pad with registered and direct input	X							
PIO	Input/output pad	X				X			
PIOC						X			
PIOQ	Input/output pad with registered input	X				X			
PIOQC						X			
POUT	Output pad	X				X			
POUTC						X			
POUTQ	Output pad with registered output	X							
POUTZ	Output pad with 3-state control	X				X			
PREG	Pad with data register	X				X			
PREG2	Pad with 2-bit shift register	X							
RESISTORS:									
PULLUP	Pullup resistor		X	X	X				

Ordering Information

Software Ordering Tree

The following chart shows how to order PGA software. A complete list of the available software is shown on the following page. Bundled packages are available, making it easier for you to select the package you need for your PGA design tasks. The diagram shown below indicates how the various packages are combined to make up the various package bundles.

AmPGA021 SW/ xx yy

Product Number
(e. g., 021 = PGA Development System)

Product Type
SW = Software
HW = Hardware
MA = Maintenance Agreement

Media Type
09 = 0.25" RBAK Cartridge
11 = 5.25" DS DD
12 = 9-trk mag tape, 1600 bpi
14 = 3.5" DS HD
22 = 3.5" DS DD
24 = 5.25" DS HD

Host
09 = Apollo
10 = AT Class
11 = XT Class
13 = PS/2 Class
14 = Daisy DNIX

Common Designators for xxyy	
System	**xxyy**
IBM PC-AT or compatible	1024
IBM PS/2	1314
Apollo with cartridge tape	0909
Apollo with magnetic tape	0912
Daisy	1424

Available Development Tools

Bundled software systems:

AmPGA151	PGA Bundled Development System (Includes AmPGA021, AmPGA023, and AmPGA135)
AmPGA251	PGA Bundled Development System with Simulation (Includes AmPGA021, AmPGA023, AmPGA135, and AmPGA145)

Development software:

AmPGA021	PGA Design System
AmPGA023	Automatic Design Implementation

Schematic Capture software:

AmPGA135	OrCAD Schematic Capture, PGA Interface and Library (Includes OrCAD/SDT III and AmPGA035)
AmPGA035	OrCAD Schematic Capture Interface and PGA Library
AmPGA031	FutureNet Schematic Capture Interface and PGA Library

Simulation software:

AmPGA145	OrCAD Simulation, PGA Interface and Library (Includes OrCAD/VST and AmPGA045)
AmPGA045	OrCAD Simulation Interface and PGA Library
AmPGA022	PC-SILOS Simulator

Workstation Support software:

AmPGA133	Daisy Schematic Capture and Simulation Interfaces and PGA Library
AmPGA134	Mentor Schematic Capture and Simulation Interfaces and PGA Library

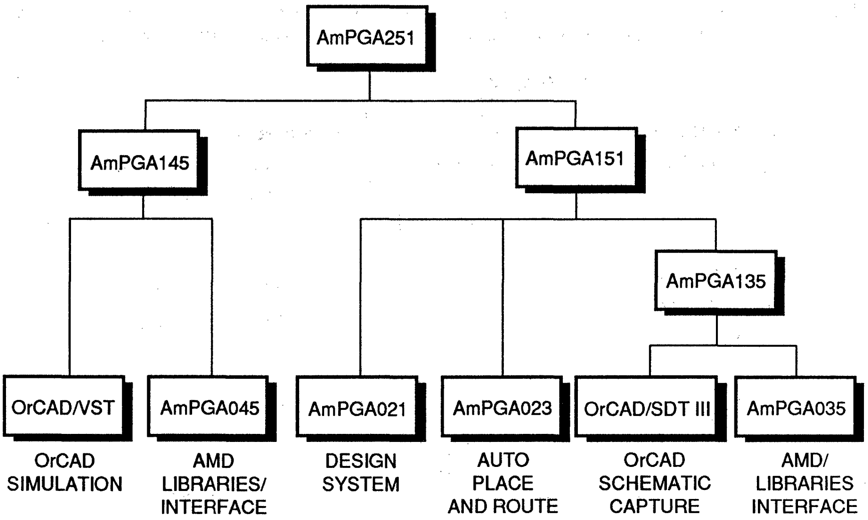
In-Circuit Emulation:

AmPGA026	In-Circuit Emulator Universal Emulation Pod
AmPGA027	In-Circuit Emulator Emulation Header
AmPGA028	In-Circuit Emulator Emulation Controller Box

Programmer:

AmPGA081	Serial Configuration PROM Programmer

Bundled Software Components



Minimum System Requirements

PC-Based software

- IBM PC-AT or 100% compatible
- MS-DOS Version 3.0 or higher
- 20 megabytes of available hard disk space
- 640 kilobytes base RAM memory
- Up to 6 megabytes RAM extended memory

Device design	**Memory required**
2064/2018/3020	2.0 Mbytes total (base + extended)
3030	3.0 Mbytes
3042	3.5 Mbytes
3064	5.0 Mbytes
3090	6.0 Mbytes

- Mouse - Mouse Systems, PC Mouse, Logitech, or compatible
- EGA or VGA color video adapter and display

Software Updates

AMD continues to make available to all users of PGA Development Software improvements and enhancements. Updates are provided free to all users who return their registration cards for one year following purchase. After that, you may purchase a Software Maintenance Agreement, to ensure continual updating of your software.

Sales Offices

North American

ALABAMA	(205) 882-9122
ARIZONA	(602) 242-4400
CALIFORNIA,	
Culver City	(213) 645-1524
Newport Beach	(714) 752-6262
Roseville	(916) 786-6700
San Diego	(619) 560-7030
San Jose	(408) 452-0500
Woodland Hills	(818) 992-4155
CANADA, Ontario,	
Kanata	(613) 592-0060
Willowdale	(416) 224-5193
COLORADO	(303) 741-2900
CONNECTICUT	(203) 264-7800
FLORIDA,	
Clearwater	(813) 530-9971
Ft. Lauderdale	(305) 776-2001
Orlando (Casselberry)	(407) 830-8100
GEORGIA	(404) 449-7920
ILLINOIS,	
Chicago (Itasca)	(312) 773-4422
Naperville	(312) 505-9517
KANSAS	(913) 451-3115
MARYLAND	(301) 796-9310
MASSACHUSETTS	(617) 273-3970
MICHIGAN	(313) 347-1522
MINNESOTA	(612) 938-0001
NEW JERSEY,	
Cherry Hill	(609) 662-2900
Parsippany	(201) 299-0002
NEW YORK,	
Liverpool	(315) 457-5400
Poughkeepsie	(914) 471-8180
Rochester	(716) 272-9020
NORTH CAROLINA	(919) 878-8111
OHIO,	
Columbus (Westerville)	(614) 891-6455
Dayton	(513) 439-0470
OREGON	(503) 245-0080
PENNSYLVANIA	(215) 398-8006
SOUTH CAROLINA	(803) 772-6760
TEXAS,	
Austin	(512) 346-7830
Dallas	(214) 934-9099
Houston	(713) 785-9001

International

BELGIUM, Bruxelles	TEL	(02) 771-91-42
	FAX	(02) 762-37-12
	TLX	846-61028
FRANCE, Paris	TEL	(1) 49-75-10-10
	FAX	(1) 49-75-10-13
	TLX	263282F
WEST GERMANY,		
Hannover area	TEL	(0511) 736085
	FAX	(0511) 721254
	TLX	922850
München	TEL	(089) 4114-0
	FAX	(089) 406490
	TLX	523883
Stuttgart	TEL	(0711) 62 33 77
	FAX	(0711) 625187
	TLX	721882
HONG KONG,	TEL	852-5-8654525
Wanchai	FAX	852-5-8654335
	TLX	67955AMDAPHX
ITALY, Milan	TEL	(02) 3390541
		(02) 3533241
	FAX	(02) 3498000
	TLX	843-315286
JAPAN,		
Kanagawa	TEL	462-47-2911
	FAX	462-47-1729
Tokyo	TEL	(03) 345-8241
	FAX	(03) 342-5196
	TLX	J24064AMDTKOJ
Osaka	TEL	06-243-3250
	FAX	06-243-3253

International *(Continued)*

KOREA, Seoul	TEL	822-784-0030
	FAX	822-784-8014
LATIN AMERICA,		
Ft. Lauderdale	TEL	(305) 484-8600
	FAX	(305) 485-9736
	TLX	5109554261 AMDFTL
NORWAY, Hovik	TEL	(03) 010156
	FAX	(02) 591959
	TLX	79079HBCN
SINGAPORE	TEL	65-3481188
	FAX	65-3480161
	TLX	55650 AMDMMI
SWEDEN,		
Stockholm	TEL	(08) 733 03 50
(Sundbyberg)	FAX	(08) 733 22 85
	TLX	11602
TAIWAN	TEL	886-2-7213393
	FAX	886-2-7723422
	TLX	886-2-7122066
UNITED KINGDOM,		
Manchester area	TEL	(0925) 828008
(Warrington)	FAX	(0925) 827693
	TLX	851-628524
London area	TEL	(0483) 740440
(Woking)	FAX	(0483) 756196
	TLX	851-859103

North American Representatives

CANADA	
Burnaby, B.C.	
DAVETEK MARKETING	(604) 430-3680
Calgary, Alberta	
DAVETEK MARKETING	(403) 291-4984
Kanata, Ontario	
VITEL ELECTRONICS	(613) 592-0060
Mississauga, Ontario	
VITEL ELECTRONICS	(416) 676-9720
Lachine, Quebec	
VITEL ELECTRONICS	(514) 636-5951
IDAHO	
INTERMOUNTAIN TECH MKTG, INC	(208) 888-6071
ILLINOIS	
HEARTLAND TECH MKTG, INC	(312) 577-9222
INDIANA	
Huntington - ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3450
Indianapolis - ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3450
IOWA	
LORENZ SALES	(319) 377-4666
KANSAS	
Merriam – LORENZ SALES	(913) 384-6556
Wichita – LORENZ SALES	(316) 721-0500
KENTUCKY	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
MICHIGAN	
Birmingham - MIKE RAICK ASSOCIATES	(313) 644-5040
Holland – COM-TEK SALES, INC	(616) 399-7273
Novi – COM-TEK SALES, INC	(313) 344-1409
MISSOURI	
LORENZ SALES	(314) 997-4558
NEBRASKA	
LORENZ SALES	(402) 475-4660
NEW MEXICO	
THORSON DESERT STATES	(505) 293-8555
NEW YORK	
East Syracuse – NYCOM, INC	(315) 437-8343
Woodbury – COMPONENT CONSULTANTS, INC	(516) 364-8020
OHIO	
Centerville – DOLFUSS ROOT & CO	(513) 433-6776
Columbus – DOLFUSS ROOT & CO	(614) 885-4844
Strongsville – DOLFUSS ROOT & CO	(216) 238-0300
PENNSYLVANIA	
DOLFUSS ROOT & CO	(412) 221-4420
PUERTO RICO	
COMP REP ASSOC, INC	(809) 746-6550
UTAH, R^2 MARKETING	(801) 595-0631
WASHINGTON	
ELECTRA TECHNICAL SALES	(206) 821-7442
WISCONSIN	
HEARTLAND TECH MKTG, INC	(414) 792-0920

Advanced Micro Devices reserves the right to make changes in its product without notice in order to improve design or performance characteristics. The performance characteristics listed in this document are guaranteed by specific tests, guard banding, design and other practices common to the industry. For specific testing details, contact your local AMD sales representative. The company assumes no responsibility for the use of any circuits described herein.

Advanced Micro Devices, Inc. 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088, USA
Tel: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450
APPLICATIONS HOTLINE TOLL FREE: (800) 222-9323 • (408) 749-5703

8/9/89
Printed in USA

ADVANCED MICRO DEVICES, INC.
901 Thompson Place
P.O. Box 3453
Sunnyvale,
California 94088-3453
(408) 732-2400
TWX: 910-339-9280
TELEX: 34-6306
TOLL-FREE
(800) 538-8450

APPLICATIONS HOTLINE
(800) 222-9323
(408) 749-5703

Printed in USA
10869B