405GP

Preliminary Application Note

PowerPC 405GP PCI Bus Boot Mode

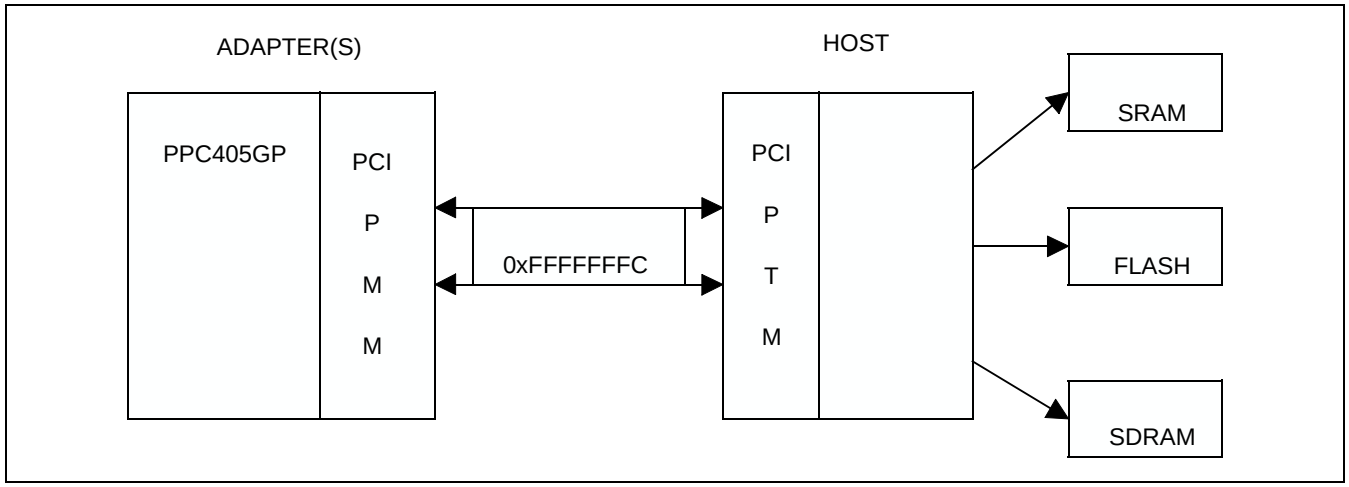
Abstract – The AMCC PowerPC™405GP (405GP) embedded processor has a mode which enables the 405GP or multiple 405GP's to access boot code located in PCI address space. Use of the PCI Boot Mode could, in some circumstances, eliminate the need for a locally attached boot ROM. The initialization/boot code can be located on or off the card on which the 405GP resides. For the purposes of this document, **adapter** refers to the 405GP that is accessing code via the PCI interface and **host** refers to the processor that controls the adapter startup and provides access to the boot code located in PCI address space.

OVERVIEW

The PowerPC 405GP PCI interface core has a mode that enables the 405 core to access PCI address space as a Processor Local Bus (PLB) master without requiring initial configuration cycles to the PCI core. System designers may use this mode to allow one or more 405GP microcontrollers to access and execute initialization/boot code that is located in PCI memory space. This mode is referred to as the PCI Boot Mode in this application note, and is enabled using a hardware-strapping pin on the 405GP.

The following simplified block diagram shows the address path for a 405GP adapter accessing boot code on a host-attached device at reset. The host attached boot device can be located anywhere in the host address space that can be accessed by the host PCI. The host processor must program the PCI Target Map (PTM) so that incoming adapter PCI addresses in the range of 0xFFFE0000 to 0xFFFFFFFF are mapped to the device containing the initialization/boot code.

Figure 1. 405GP Adapter Address Path Block Diagram



HARDWARE STRAPPING REQUIREMENTS

The *ROM Location* strapping pin on the adapter 405GP must be strapped high during reset to enable the PCI boot mode. Refer to the Strapping section of the 405GP Data Sheet to locate the correct pin. This hardware strap causes the adapter PCI address space to be mapped at 0xFFFE0000. With *ROM location* strapped high, the 405GP PCI interface comes out of reset with PCI Master Map 0 (PMM0) enabled and programmed for the address range 0xFFFE0000 œ> 0xFFFFFFFF. The Master Enable (ME) bit in the PCI Command register (PCICMD[ME]) is also set to 1 after reset. The ME bit enables the adapter 405GP PCI interface to initiate master cycles on the PCI bus. Note that enabling PCI boot mode does not prevent subsequent updates to the PMM0 registers.

SOFTWARE REQUIREMENTS

After system reset with PCI Boot Mode enabled, the 405GP fetches the first instruction from 0xFFFFFFFC as it normally does. This requires that the instruction the 405GP accesses at 0xFFFFFFFC be a hard branch to executable initialization code in the address range from 0xFFFE0000 to 0xFFFFFFFF. Since this is PCI address space, the host software that controls the startup of the adapter must insure that the necessary initialization code is in place. A host PTM register must point address 0xFFFxxxxx from the adapter PCI to this code before the adapter 405GP executes the first instruction following a reset. The code can reside in FLASH, SRAM, SDRAM or other storage media that the host and adapter PCI can access.

LIMITATIONS IN AN OPEN ENVIRONMENT

The host computer cannot access adapter identity information through the adapter 405GP PCI interface while the PCI interface is held in reset. Since the 405GP system reset input resets both the 405 core and the PCI interface, there is no way to take the adapter PCI interface out of reset and hold the 405 core in reset. The 405GP debugger Halt can be used to keep the 405 core from starting after system reset is removed, but there is no standard PCI interface signal defined to control this signal. This imposes some restrictions in an open/plug and play environment such as a PC adapter. In this environment, the adapter would require that some non-volatile program storage be implemented on the card such that the 405GP can start executing startup code when system reset is removed.

In a closed environment where the host operating system has prior knowledge of the adapter type, this limitation does not create a problem. The limitation is a problem only in an open environment (e.g.: Plug and Play) where the host (BIOS/Windows) has to read configuration information via PCI to determine which driver to activate.

PPC405GP RESET CONSIDERATIONS

The host processor will be required to hold the adapter 405PG in reset until the host PCI bus is configured so that the initialization/boot code is available. The following are two possible methods for controlling the adapter reset. In the first example, the signal names used assume that the host and adapter processors are 405GP.

1. Connect the SysReset pin of each adapter 405GP to PCI_RST#. Assert PCI_RST# from the host processor by setting the DPR bit in the PCI Bridge Option 2 register to a “one” (PCIC0_BRDGOPT2 [DPR]=1).
2. Connect the SysReset pin of each adapter 405GP to a separate latch that is controlled by the “host” processor. Using this method, the “host” processor can control the reset of each “adapter” processor individually.

The Halt pin or the SysReset pin on the 405GP can be used to control when the processor starts to boot. It is possible to either delay release of SysReset, or make Halt active before SysReset is de-asserted then release Halt at a later time to control the start of the 405GP boot process.

When SysReset is first de-asserted to a 405GP, the 405GP internally continues to reset itself for 8K (8192) clocks. During this time, if the host attempts to access the recently reset de-asserted adapter 405GP, the PCI cycle should master-abort, and no PCI device hangs. However, there is the possibility of a hang if the host attempts to access the adapter 405GP, as it is coming out of its internal reset. The host should not attempt to access a 405GP via the PCI interface until the 405GP has completed its reset.

When the adapter 405GP completes reset, (PCIC0_BRDGOPT2 [HCE]=1) (Host Configuration Enable) bit is active, and the 405GP begins loading boot code. All configuration cycles from the host PCI to the adapter are retried until the adapter clears the HCE bit. The host repeats retried transfers until they complete. Thus, if the host attempts a PCI configuration access to the adapter, it will not progress forward in any way until the HCE bit is cleared. To avoid this, the host should wait until the newly enabled 405GP has had time to clear the HCE bit before accessing its configuration registers.

USING RISCWATCH™ WITH PCI SPACE

Following reset with the 405GP configured to boot from PCI, the address space from 0xFFFE0000 through 0xFFFFFFFF is mapped to PCI address space. RISCWatch does not recognize this as valid address space. If you want to debug the boot code that resides in PCI address space using the RISCWatch, issue the following RISC-Watch command:

```
memacc add 0xFFFE0000 0xFFFFFFFF
```

If desired, this command can be added to a startup command file and executed each time RISCWatch is started. Execution of a startup command file is controlled by the STARTUP_FILE option in the rwppc.env file (e.g. STARTUP_FILE = startup.cmd).

405GP REV B INFORMATION

The 405GP Rev B processor will hang if the PCI device that the 405GP is booting from issues a PCI disconnect while the boot is in progress. This PCI boot problem does NOT occur when booting a 405GP adapter from 405GP host because the 405GP does not issue PCI disconnects during the PCI boot transfers. This PCI disconnect problem did not exist with the 405GP Rev A and it will be corrected in 405GP Rev C and Rev D. See 405GP Rev B errata #11 for possible work-around.

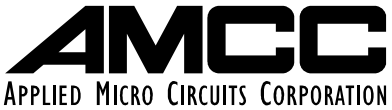
HOST SOFTWARE EXAMPLE

The following code example shows how a 405GP host PCI could be initialized to support a 405GP adapter booting from initialization/boot code stored in the host DRAM.

```
/*---------------------------------------------------------------------------+
| pci_init. Initializes the 405GP PCI Configuration regs.
+---------------------------------------------------------------------------*/
void pci_init()
{

 unsigned short temp_short;
 unsigned long temp_long;
   /*---------------------------------------------------------------------------+
| Assert PCI reset to give us time to set up PMMs PTMs for allowing another
| 405 based PCI device to boot from us.
   +---------------------------------------------------------------------------*/
   temp_short = PCI_Read_CFG_Reg(PCIDEVID_405GP, PCIBRDGOPT2, 2);
   PCI_Write_CFG_Reg(PCIDEVID_405GP, PCIBRDGOPT2, (temp_short | 0x1000), 2);

   /*---------------------------------------------------------------------------+
| 405GP PCI Master configuration.
| Map one 512 MB range of PLB/processor addresses to PCI memory space.
| PLB address 0x80000000-0x9FFFFFFF ==> PCI address 0x80000000-0x9FFFFFFF
| Use byte reversed out routines to handle endianess.
   +---------------------------------------------------------------------------*/
   out32r(PMM0LA,    0x80000000);
   out32r(PMM0PCILA, 0x80000000);
   out32r(PMM0PCIHA, 0x00000000);
   out32r(PMM0MA,    0xE0000001); /* no prefetching, and enable region */

   /*---------------------------------------------------------------------------+
| This space is for the VGA card.
```

```
  +----------------------------------------------------------------------*/
   out32r(PMM1LA,    0xA0000000);
   out32r(PMM1PCILA, 0x00000000);
   out32r(PMM1PCIHA, 0x00000000);
   out32r(PMM1MA,    0xE0000001);    /* no prefetching, and enable region */

   /*----------------------------------------------------------------------+
| PMM2 is not used.  Initialize them to zero.

  +----------------------------------------------------------------------*/
   out32r(PMM2LA,    0x00000000);
   out32r(PMM2PCILA, 0x00000000);
   out32r(PMM2PCIHA, 0x00000000);
   out32r(PMM2MA,    0x00000000); /* not enabled */

   /*----------------------------------------------------------------------+
| 405GP PCI Target configuration. (PTM1)
| Map one 2 GB range of PCI addresses to PLB/processor address space.
| PCI address 0x00000000-0x7FFFFFFF ==> PLB address 0x00000000-0x7FFFFFFF
| The 0x00000008 default value of the 405GP PTM1 Base Address Register
| (in the PCI config header) is correct for this mapping.
| Note: PTM1MS is hardwire enabled but we set the enable bit anyway.

  +----------------------------------------------------------------------*/
   out32r(PTM1LA,    0x00000000);
   out32r(PTM1MS,    0x80000001); /* 2GB, enable bit is hard-wired to 1 */

   /*----------------------------------------------------------------------+
| 405GP PCI Target configuration. (PTM2)
|
| Allows an adapter 405GP to boot from our SRAM.
|
| Addresses 0xFFF80000 - 0xFFFFFFFF from the slave processor will be mapped
| to our  0xFFF00000 - 0xFFF7FFFF
   +----------------------------------------------------------------------*/
   out32r(PTM2LA,    0xFFF00000);   /* Map to the SRAM */
   out32r(PTM2MS,    0xFFF80001); /* 512KB, set enable bit */

   temp_long = PCI_Read_CFG_Reg(PCIDEVID_405GP, PCIBASEADDR2, 4);
   temp_long |= 0xFFF80000; /* accept addresses from boot area of slave */
   PCI_Write_CFG_Reg(PCIDEVID_405GP, PCIBASEADDR2, temp_long, 4);

   /*----------------------------------------------------------------------+
| Write the 405GP PCI Configuration regs.
| Enable 405GP to be a master on the PCI bus (PMM).
| Enable 405GP to act as a PCI memory target (PTM).
   +----------------------------------------------------------------------*/
   temp_short = PCI_Read_CFG_Reg(PCIDEVID_405GP, PCICMD, 2);
   PCI_Write_CFG_Reg(PCIDEVID_405GP, PCICMD, temp_short | BM_EN | MEM_EN, 2);

   /*----------------------------------------------------------------------+
| If PCI speed = 66Mhz, set 66Mhz capable bit.
```

```
   +---------------------------------------------------------------------------*/
 if (board_cfg.pci_speed==66666666) {

                temp_short = PCI_Read_CFG_Reg(PCIDEVID_405GP, PCISTATUS, 2);
PCI_Write_CFG_Reg(PCIDEVID_405GP,PCISTATUS,(temp_short|CAPABLE_66MHZ), 2); }
   /*---------------------------------------------------------------------------+
| Default value of the Bridge Options1 register is OK (0xFF60).
| Default value of the Bridge Options2 register is OK (0x0100).
| No need to change them in pass 2 405GP.
| Low subsequent target latency timer values are not supported in pass 1.
| STLD = '1111' if asynchronous PCI
| STLD = '0111' if synchronous PCI
   +---------------------------------------------------------------------------*/

   /* The following should only be required for PASS 1 405GP because of */
   /* errata #26  */

   temp_short = PCI_Read_CFG_Reg(PCIDEVID_405GP, PCIBRDGOPT2, 2);

   /* TURN OFF PCI RESET BIT */

   temp_short &= 0xEFFF;

 if (ppcMfstrap() & PCI_ASYNC_MODE_EN) {

PCI_Write_CFG_Reg(PCIDEVID_405GP, PCIBRDGOPT2, (temp_short | 0x0F00), 2);
}
else {
PCI_Write_CFG_Reg(PCIDEVID_405GP, PCIBRDGOPT2, (temp_short | 0x0700), 2); }
```



Applied Micro Circuits Corporation
6290 Sequence Dr., San Diego, CA 92121
Phone: (858) 450-9333 — (800) 755-2622 — Fax: (858) 450-9885
http://www.amcc.com