DECEMBER 1982
VOL. 61, NO. 10, Part 1

THE BELL SYSTEM TECHNICAL JOURNAL

THE BELL SYSTEM TECHNICAL JOURNAL

THE BELL SYSTEM TECHNICAL JOURNAL (ISSN0005-8580) is published by the American Telephone and Telegraph Company, 195 Broadway, N.Y., N.Y. 10007, C. L. Brown, Chairman and Chief Executive Officer; W. M. Ellinghaus, President; V. A. Dwyer, Vice President and Treasurer; T. O. Davis, Secretary.

The Journal is published in three parts. Part 1, general subjects, is published ten times each year. Part 2, Computing Science and Systems, and Part 3, single-subject issues, are published with Part 1 as the papers become available.

The subscription price includes all three parts. Subscriptions: United States—1 year $35; 2 years $63; 3 years $84; foreign—1 year $45; 2 years $73; 3 years $94. Subscriptions to Part 2 only are $10 ($11 Foreign). Single copies of the Journal are available at $5 ($6 foreign). Payment for foreign subscriptions or single copies must be made in United States funds, or by check drawn on a United States bank and made payable to The Bell System Technical Journal and sent to Bell Laboratories, Circulation Dept., Room 1E-335, 101 J. F. Kennedy Parkway, Short Hills, N. J. 07078.

Comments on the technical content of any article or brief are welcome. These and other editorial inquiries should be addressed to the Editor, The Bell System Technical Journal, Bell Laboratories, Room 1J-319, 101 J. F. Kennedy Parkway, Short Hills, N. J. 07078. Comments and inquiries, whether or not published, shall not be regarded as confidential or otherwise restricted in use and will become the property of the American Telephone and Telegraph Company. Comments selected for publication may be edited for brevity, subject to author approval.

Printed in U.S.A. Second-class postage paid at Short Hills, N.J. 07078 and additional mailing offices. Postmaster: Send address changes to The Bell System Technical Journal, 101 J. F. Kennedy Parkway, Short Hills, N.J. 07078

THE BELL SYSTEM
TECHNICAL JOURNAL

DEVOTED TO THE SCIENTIFIC AND ENGINEERING
ASPECTS OF ELECTRICAL COMMUNICATION

Volume 61 December 1982 Number 10

An Overview of PANACEA, a Software Package for Analyzing Markovian Queueing Networks

By K. G. RAMAKRISHNAN and D. MITRA

(Manuscript received April 19, 1982)

PANACEA is a software package that significantly extends the range of Markovian queueing networks that are computationally tractable. It solves multi-class closed, open, and mixed queueing networks. Based on an underlying theory of integral representations and asymptotic expansions, PANACEA solves queueing networks that are orders of magnitude larger than can be solved by other established algorithms. The package is finding widespread use in Bell Laboratories. It also has important software innovations. A flexible programming-language-like interface facilitates compact representation of large queueing networks. An out-of-core implemental strategy enables PANACEA to be ported to processors with modest memory. The modular structure of this software package, along with the automatic machine-generated parser, makes it easily extendable. This paper provides an overview of two basic versions of PANACEA, versions 1.0 and 1.1, which solve "closed" networks only. A description of its model language is given from the point of view of its capability to describe queueing networks in a compact, natural manner. The paper discusses the algorithms, together with their time and storage requirements, that are used in the implementation. Several numerical examples are given.

I. INTRODUCTION

Closed Markovian queueing networks are acknowledged to be valuable tools for analyzing computer systems, computer communication systems, on-line computer networks, and other real-time computer-based systems (Refs. 1 through 12 document many real-life examples). Despite this large collection of problems modeled by closed queueing networks, until recently, only a small number of these models were computationally tractable using established algorithms.[13-17] The tractable models have small numbers of customer classes and a small population in each customer class. PANACEA (Package for Analysis of Networks of Asynchronous Computers with Extended Asymptotics) extends the class of computationally tractable, closed queueing networks by several orders of magnitude.

The solution of "large" queueing networks by PANACEA is made possible by an underlying theory of integral representations and asymptotic expansions of relevant performance measures.[18-20] PANACEA obtains solutions to queueing networks with p customer classes and q processing centers, in $O(qp^t)$ operations and about $20q$ kilobytes of storage. Here "t" denotes the number of terms in the asymptotic expansions of the first and second moments of quantities of interest (utilizations, queue lengths, etc.). Extensive computational experiments demonstrate that four terms ($t = 4$) are usually more than adequate to produce highly accurate results. Most importantly, the number of operations and the amount of storage are both independent of the population sizes of customer classes. In contrast, both the convolutional algorithm and mean value analysis[13-17] require operations and storage proportional to

$$p(1 + 2q) \prod_{j=1}^{p} K_j,$$

where K_j is the population size of customer class j.

Thus, PANACEA is distinguished by its ability to solve exponentially growing problems in polynomial time. There are additional important features to PANACEA:

(*i*) PANACEA incorporates a special-purpose queueing network language designed to describe queueing networks.

(*ii*) PANACEA computes second moments of queue lengths. We know of no other package that provides this information.

(*iii*) All computed measures of network performance are typically accompanied by upper and lower bounds.

(*iv*) A complete complexity count enables PANACEA to guarantee a response time for a problem solution by automatically selecting the maximum number of terms to use in the asymptotic expansions. This, in conjunction with (*iii*) above, is often useful in the early stages of system design.

PANACEA runs interactively on computer systems with a C-language compiler and LEX,[21] the automated generator of lexical analyzer. This paper describes two basic versions, 1.0 and 1.1, of the package. Other versions have subsequently been implemented and these will be described elsewhere. Version 1.0 solves closed queueing networks in which customers are not allowed to change classes while circulating in the network (no class hopping). This version is implemented completely in-core. Version 1.1 provides for the "class-hopping" feature of queueing networks. An analysis of class-hopping queueing networks shows very large storage requirements even for networks of moderate size. Hence, in Version 1.1 an out-of-core strategy is employed.

Versions 1.0 and 1.1 each consist of about 6000 lines of portable C code, and currently run on VAX 11/780 hardware. Queueing network problems are solved in a guaranteed response time of at most one minute on a nominally utilized VAX 11/780. Many experiments conducted on PANACEA indicate that it is robust and numerically stable. We should mention that PANACEA is being used currently in several projects within Bell Laboratories. The experiences of the users are guiding the enhancements to PANACEA.

The interface to both versions of PANACEA has been designed to allow the user to describe large queueing networks compactly and in a natural manner. Many of the features of general-purpose programming languages have been incorporated into PANACEA Model Language (referred to as PML, henceforth), thus making it a special-purpose queueing network language. Features of PML, such as the preprocessor variable usage, free use of comments, etc., have been freely borrowed from programming languages.

PANACEA is envisioned to run on hardwares ranging from microprocessors to large mainframes. The expressions derived for time and storage requirements in Section IV enable the user to "package" PANACEA to suit the availability of computing resources such as central processing unit (CPU) cycles and memory. For example, in Motorola 68000 with a 0.2 million instruction per second and 256K physical user memory, the user can solve network problems as large as one in which there are 12 processing centers and 10 job classes, in a guaranteed response time of at most one minute. [See eqs. (15) and (16).]

We mention some of the main limitations of PANACEA in its present form. The queueing networks are required to belong to the class of "product form" networks.[1,4] In addition, "normal usage" must be operative in these networks. Normal usage is defined in Section 3.1. In practice this translates to the requirement that no processing center in the queueing network is utilized more than about 85 percent. Finally, load-dependent service rates are not allowed in the existing versions of PANACEA.

Our discussion of PANACEA is organized into six sections. Section II describes the global properties of PML and the overall software structure of PANACEA. Section III details the internals of the computational phase, together with an overview of the theory of asymptotic expansions. Section IV, "Computational Complexity," counts operations and storage required, and also gives experimental results for validation. Several numerical examples illustrating different features of PANACEA are given in Section V. Section VI presents conclusions and proposed future enhancements to PANACEA.

II. FEATURES OF PML

This section presents an overview of PML. Rigorous syntax and semantic definitions of PML statements have been developed.[22] Properties of PML statements are illustrated by considering a prototypical queueing network shown in Fig. 1. This queueing network represents a service point of a large packet-switched communications network. This service point consists of three front-end processors (terminal concentrators) that act as gateways for the terminals connected to the service point. The front-end processors (FEPs) are connected to three central processing units (CPUs). These processors, when in need of data base access, communicate with a data base processor. The labels within the processors are symbols used in the PML program describing the queueing network. These labels are arbitrary and can be chosen by the user in any manner. The node labeled TERMINAL represents the terminal population connected to the service point. This terminal population is divided into nine customer classes. After obtaining service at any of the CPUs, customer classes 1 through 4 return to the TERMINAL to generate the next command, whereas customer classes 5 to 9 transit to DBP, since they require data base access. Figure 2 gives the PML program that describes the queueing network shown in Fig. 1. The lines in the program are numbered for easy reference to statements. They are not part of the PML program. The following discussion of PML should be read in conjunction with Fig. 2.

2.1 Overview

PML is a free-format language. Line boundaries, tabs, and white spaces in the statements are completely ignored. Users are permitted the flexibility of segmenting statements across multiple lines, indenting portions of their statements, specifying multiple statements on a line, or using a combination of the above. A statement in PML describes some aspect of the queueing network being solved. For example, the statement on line number 6 in Fig. 2 specifies the routing probabilities between TERMINAL and the FEPs. PML also provides the flexibility of symbolic representation of any entity of the queueing network,

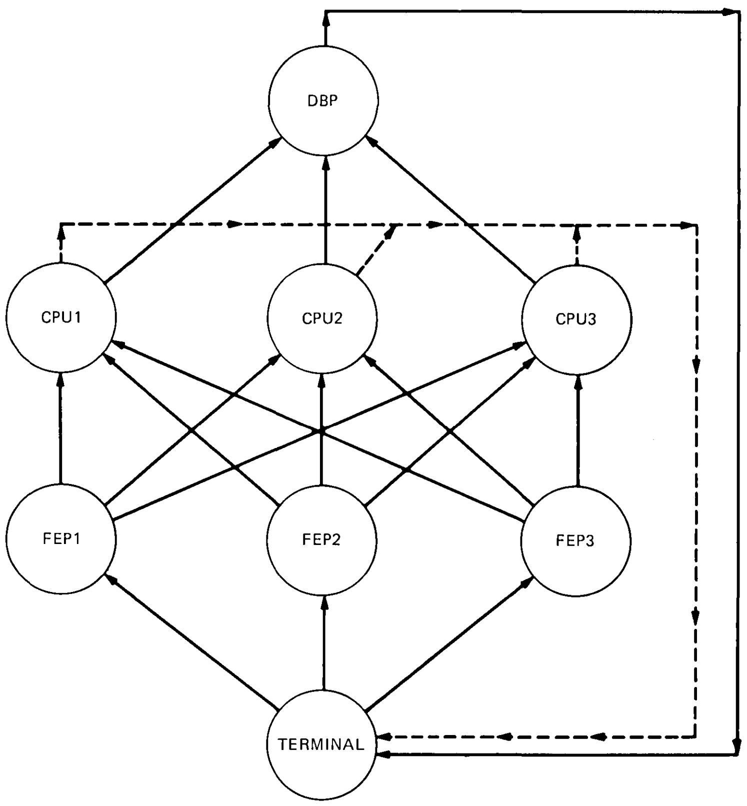


Fig. 1—Model example.

using symbols of arbitrary length. A symbol table manager in the parser module of PANACEA manages these symbols specified by the PML program. Symbols of the PML program become contextually defined as to their data type when they appear in the program. No statements are needed in PML to declare data types for symbols. For example, the symbols "total-classes" and "call-classes" in lines 3 and 4 of Fig. 2 become contextually defined as preprocessor variables of type "integer." Similarly, the symbols "TERMINAL," "FEP1," "FEP2," and "FEP3"—appearing in line 6 of Fig. 2—acquire the data type "string." PML allows arbitrarily complex arithmetic expressions to be specified in the statements. The arithmetic operators in these expressions have the standard precedence and associativity. PML programs can be made readable and self-documenting by making use of the facility for comments embedded in the program. With this general introduction to PML, we will now attempt to describe some of its global properties.

```
 1 /*ILLUSTRATIVE EXAMPLE*/
 2 /*PREPROCESSOR DEFINITIONS*/
 3                          total_classes = 9;
 4                           call_classes = 4;
 5 /*ROUTING DESCRIPTIONS*/
 6 TERMINAL        :FEP1,FEP2,FEP3  1.0/3.0;
 7 FEP1,FEP2,FEP3 :CPU1,CPU2,CPU3   1.0/3.0;
 8 CPU1,CPU2,CPU3:TERMINAL
 9                 {(1 to call_classes) 1.0};
10 CPU1,CPU2,CPU3:DBP
11                 {(call_classes + 1
12                    to
13                  total_classes) 1.0};
14 DBP             :TERMINAL
15                 {(call_classes + 1 to
16                  total_classes) 1.0};
17 /*RATE DESCRIPTIONS*/
18 CPU1,CPU2,CPU3 {12.5,20,12.5,20,2.5,3.3,2.5,3.3,8.33};
19 'FEP.*'         10.0;
20 DBP {(call_classes+1).73611,.73611,.3833,.3823,3.33};
21 TERMINAL {.00333,.00444,.0008333,.01555,
22               .0008333,.0008335,.000076,.000079,.0083};
23 /*DISCIPLINE DEFINITIONS*/
24 CPU1,CPU2,CPU3 PS;TERMINAL IS;DBP PS;
25 FEP1,FEP2,FEP3 FCFS;
26 DEGREE {(1 to 4) 75,150,150,5,5,180};
27 OUTPUT DBP UTIL + QLENGTH + Q2LENGTH;
28 END;
```

Fig. 2—PML program.

2.2 Global properties

PML is characterized by the global properties of compactness, extendability and self-documentation, as discussed below.[23]

2.2.1 Compactness

A programming language is considered compact if the user can specify "aggregate" entities using a single "atomic" entity descriptor. PML enables the user to specify large networks compactly. Many features of the language aid in this compactness.

2.2.1.1 Multiple edge and node descriptors. All entities in the network that have similar characteristics are described together in a single statement. For example, the statement in line 7 of Fig. 2 states that the transition probability for *all edges* emanating from the front-end processors and terminating in the node processors is 1/3. Thus, 81 edges are described in a single statement, as explained below.

The node symbols separated by commas and delimited by ":" denote the set of originating nodes. The comma-separated node symbols specified after the ":" denote the set of destination nodes. Thus, in

Fig. 2, every origin-destination pair specifies nine edges (one for each customer class). As a universal rule in PML, aggregate names (node names separated by commas) can be given wherever a single node name is expected. The operand specification is associated with every node in the aggregate name. For instance, the service rate descriptor in line 18 of Fig. 2 specifies all the service rates of the CPUs in a single statement.

2.2.1.2 Implicit class propagation. When many customer classes have identical characteristics, the user specifies the characteristics exactly once. All customer classes implicitly obtain those characteristics, thus resulting in compactness. If we refer back to line 7 in Fig. 2 again, we can see that the transition probability 1/3 specified in the operand propagates to all nine customer classes. If the customer classes differ in their characteristics, then a separate value can be specified for each class, as line 18 shows. The point to note here is the implicit class indexing agreed upon between the parser and the PML programmer. Another feature of PML allows the user to group some classes that have identical characteristics by prefixing the operand value by a group range, as we see in lines 9, 15, and 16. In line 9, the expression enclosed in angle brackets "⟨" and "⟩" specifies the class range to be 1 to 4 (recall that call-classes = 4), and a value of 1.0 is assigned to the probability of transition for these classes. The class indices outside the range are unaffected. Thus, the user has the flexibility of enumerating each class, grouping classes, or omitting the specification for the classes altogether.

2.2.1.3 MACROS, INCLUDE FILES. and regular expressions. PML provides many language tools to condense frequently occurring sequences of statements. MACROS and INCLUDE FILES enable PML programs to be compact. PML MACROS and INCLUDE FILES follow standard conventions of MACRO and INCLUDE FILE specification of programming languages. Another tool that promotes compactness is regular expressions. Any collection of symbols having common subsymbols can be specified by a single regular expression. For instance, the regular expression 'FEP.*' in line 19 of Fig. 2 is a regular expression that specifies the three nodes FEP1, FEP2, and FEP3.

2.2.2 Extendability

The PML parser and lexical analyzer have been machine generated. Once PML grammar rules and lexical tokens are specified, the parser and the lexical analyzer can be generated by existing automated tools (Yacc[24] and Lex[21]). This implies that it is trivial to extend or modify the language by simply respecifying the grammar rules and/or the lexical tokens. PANACEA is designed to be an integrated queueing network package with the ability to handle most classes of queueing networks. The extendability feature of PML is crucial in enhancing the descriptive nature of PML, to describe, for instance, open networks, priority networks, etc.

2.2.3 Self-documentation

PML aids in self-documenting the model, by the use of comments anywhere in the model, preprocessor variables, and indentation and structuring of the model. We have deliberately omitted details of each statement in the PML program of Fig. 2, in the hope that the self-documenting feature of PML will obviate this necessity.

2.3 Structure of PANACEA

Figure 3 depicts the structure of PANACEA software. The input phase of PANACEA consists of the lexical analyzer and the parser. These two modules work in lockstep, parsing statements of the user program (written in PML), validating the statements, and gleaning the essential information from these statements. The object module that is created is then consumed by the control module, which is table driven. The asymptotic series for each quantity of interest (utilization, mean queue length, second moments on queue length, etc.) is encoded in a table. The table contains information on how to compute the elements of the series in terms of the moment partition functions of the pseudo-networks (see Section III). These pseudo-networks are solved by the pseudo-network modules using recursive techniques. Finally, all relevant quantities of interest requested by the user are displayed as output by the output module. PANACEA has been modularly structured in this fashion, so that advanced users can bypass one or more of the three phases of PANACEA (see Fig. 3), thus making the execution faster. For example, an object module that has been previously created can be directly passed on to the computation phase, bypassing the compilation phase.

The above discussion is relevant to both Versions 1.0 and 1.1. However, Version 1.1, which implements the class-hopping feature of queueing networks, has an additional feature. The storage of the routing matrices and the process of solving for their Perron eigenvector are done out-of-core. A "MAP" is kept in-core denoting the exact position, in secondary storage, of the matrix element $P_{(\sigma,\tau)(j,i)}$ (transition probability of going from node τ to node i while hopping from customer class σ to customer class j). Whenever this element needs to be retrieved or stored, PANACEA performs an input/output operation to the appropriate place in secondary storage. Naturally, a compromise is made in sacrificing run time while gaining significant reduction in virtual memory. The run time can be improved by "optimal" caching of matrix elements, according to the availability of physical memory.

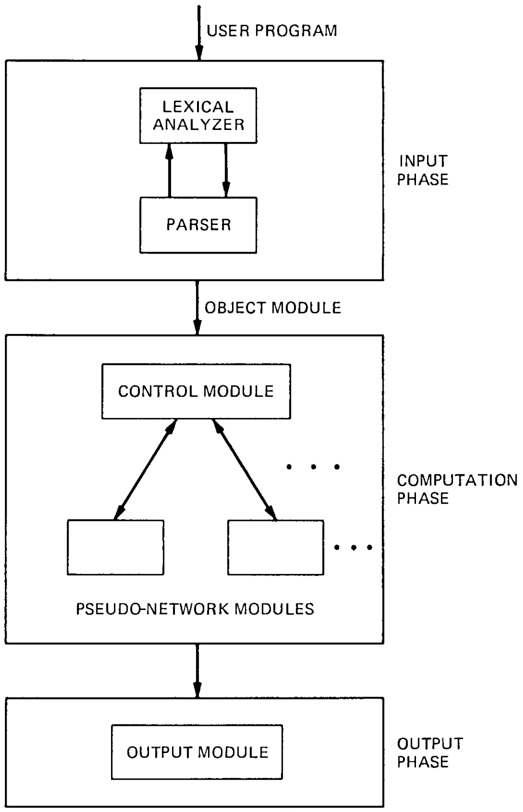


Fig. 3—Structure of PANACEA.

III. THE COMPUTATIONS

3.1 Background to asymptotic expansions

Our description of the background to asymptotic expansions shall be brief. We begin with the main results arrived at in the companion papers.[18-20] In addition, we restrict our descriptions to the first and autocorrelative second moments of individual queue lengths in the processors; various other quantities, such as throughput and processor utilization, are also computed in PANACEA but omitted in the discussion as they are either simply derived from or closely related to the quantities discussed.

In the network

$$p = \text{number of classes of jobs,} \tag{1}$$

$$q = \text{number of active (i.e., of types 1, 2, and 4) processing centers}^4 \tag{2}$$

and let the classes and centers be indexed by σ and τ, respectively. Let $n_{\sigma\tau}$ denote the random number of jobs of class σ in center τ. For the leading moments from the stationary distributions of individual processor queue lengths, we have from Refs. (19) and (20) the following *exact* expressions:

$$\left.\begin{aligned} \langle n_{\sigma\tau}\rangle(\mathbf{K}) &= \frac{r_{\sigma\tau}K_\sigma}{\alpha_\tau}\frac{I^{(1)}_{\sigma,\tau}(N)}{I(N)} \\ \langle n_{\sigma\tau}(n_{\sigma\tau}-1)\rangle(\mathbf{K}) &= \frac{r^2_{\sigma\tau}K_\sigma(K_\sigma-1)}{\alpha^2_\tau}\frac{I^{(2)}_{\sigma,\tau}(N)}{I(N)} \end{aligned}\right\} \begin{aligned} &1 \le \sigma \le p, \\ &1 \le \tau \le q \end{aligned} \qquad (3)$$

where

K_σ = population of jobs of class σ

$r_{\sigma\tau} = \rho_{\sigma\tau}/\rho_{\sigma 0}$

$$\rho_{\sigma\tau} = \frac{\text{relative number of visits of class } \sigma \text{ jobs to center } \tau}{\text{service rate of class } \sigma \text{ jobs in center } \tau}$$

$$\rho_{\sigma 0} = \sum\left[\frac{\text{relative number of visits of class } \sigma \text{ jobs to center } \tau}{\text{service rate of class } \sigma \text{ jobs in center } \tau}\right],$$

where the sum is over all infinite server centers τ visited by class σ jobs;

$$\alpha_\tau = 1 - \sum_\sigma K_\sigma r_{\sigma\tau} \ (>0 \text{ in "normal usage"})$$

$I^{(m)}_{\sigma,\tau}(N)$ = integrals parameterized by N

$m = 0, 1, 2; \quad 1 \le \sigma \le p; \quad 1 \le \tau \le q$

$I(N) = I^{(0)}_{\sigma,\tau}(N)$ (for $m = 0$ there is no dependence on σ or τ)

N = large parameter.

PANACEA chooses

$$N = 1/\min_{\sigma,\tau} r_{\sigma\tau}. \qquad (4)$$

While this choice is not at all critical, theoretical reasons corroborated by computational experience indicate that it serves very well to keep well within machine range all terms calculated in the expansions to be described below.

Thus, the quantities requiring computation are the integrals and the theory has developed the asymptotic expansions

$$I^{(m)}_{\sigma,\tau}(N) \sim \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k. \qquad (5)$$

Let us digress on

$$t = \text{number of terms in asymptotic expansion.} \qquad (6)$$

The error in the calculation of the integrals, and therefore of the queue length moments in (3), from using only t terms is $0(1/N^t)$. In PANACEA, generally,

$$t \leq 4. \qquad (7)$$

[For $I(N)$ only, the maximum number of terms is 5.] However, PANACEA has the facility to automatically select t to be less than the maximum. There are two reasons for this. First, if the user desires a guaranteed response time, our detailed complexity count (see Section IV) allows PANACEA to satisfy this by calculating the appropriate value of t. As all output quantities are typically accompanied by lower and upper bounds (see Section 3.6), this facility leading to small t is both useful and often exercised. A second reason is that PANACEA looks for departure from monotonicity of the series in (5), and in the event of its occurrence it truncates the series optimally. This a rare phenomenon and always occurs, as the theory explains, for small networks when N given in (4) is not large enough.

The following resumes the discussion on the computation of the expansion coefficients $A^{(m)}_{\sigma,\tau,k}$.

3.2 The pseudo-networks in the computation of the expansion coefficients

The expansion coefficients $\{A^{(m)}_{\sigma,\tau,k};\ 0 \leq m \leq 2,\ 1 \leq \sigma \leq p,\ 1 \leq \tau \leq q,\ 0 \leq k \leq t-1\}$ have been completely specified in Refs. 19 and 20 as simple algebraic combinations of other basic quantities $g^{(m)}_\tau(\mathbf{n})$. For example,

$$\begin{aligned} A^{(1)}_{\sigma,\tau,2} = {} & 2 \sum_j \beta_j g^{(1)}_\tau(3\mathbf{e}_j) + \frac{1}{8} \sum_j \beta_j^2 g^{(1)}_\tau(4\mathbf{e}_j) \\ & + \frac{1}{2} \sum_{j,k} \beta_j \beta_k g^{(1)}_\tau(2\mathbf{e}_j + 2\mathbf{e}_k) + 3\beta_\sigma g^{(1)}_\tau(3\mathbf{e}_\sigma) \\ & + \sum_{j \neq \sigma} \beta_j g^{(1)}_\tau(\mathbf{e}_\sigma + 2\mathbf{e}_j) + 2g^{(1)}_\tau(2\mathbf{e}_\sigma), \end{aligned} \qquad (8)$$

in which j and k are distinct class indices, $\mathbf{e}_j$ is the vector with 1 in the jth location and 0 elsewhere, and $\beta_j = K_j/N$. (For $m = 0$, $A^{(m)}_{\sigma,\tau,k}$ is independent of σ, τ and also denoted by A_k.)

It turns out that the underlying quantities $g^{(m)}_\tau(\mathbf{n})$ are the mth moment partition functions[20] of a hypothetical network with population vector $\mathbf{n}$. This latter network is related to the original network in having the same number, q, of processing centers but has no infinite server center and has quite different processing rates. The two important features to note of the population vectors $\mathbf{n}$ that appear as arguments of $g_\tau^{(m)}$ in formulas such as (8) are that

$$\text{at most } t \text{ classes have nonzero population,} \tag{9a}$$

$$\text{the total population, } \sum n_\sigma \leq 8. \tag{9b}$$

For the worst case $t = 4$, we represent

$$\mathbf{n} = n_i\mathbf{e}_i + n_j\mathbf{e}_j + n_k\mathbf{e}_k + n_\ell\mathbf{e}_\ell, \qquad n_i + n_j + n_k + n_\ell \leq 8, \tag{10}$$

where (i, j, k, ℓ) are class indices of the original network and thus $1 \leq i, j, k, \ell \leq p$. Without loss of generality we consider only $1 \leq i < j < k < \ell \leq p$.

For a particular choice of a 4-tuple (i, j, k, ℓ) we therefore have a hypothetical network with only four classes. We call such a restricted hypothetical network a pseudo-network. Clearly, there are as many as $\binom{p}{4}$ pseudo-networks in the worst case and $\binom{p}{t}$ in general. However, each pseudo-network is small.

To summarize, PANACEA computes t $(t \leq 4)$ terms in the expansions of each of $(1 + 2pq)$ integrals, and the expansion coefficients $A_{\sigma,\tau,k}^{(m)}$ are computed from the moment partition functions of the pseudo-networks of which there are $\binom{p}{t}$, each with t classes and a total population over all classes of at most 8.

3.3 The pseudo-network computations

PANACEA solves for the mth moment partition functions $\{g_\tau^{(m)}(\mathbf{n})\}$ for each of the pseudo-networks in turn where each pseudo-network is characterized by a leading t-tuple from (i, j, k, ℓ), $1 \leq i < j < k < \ell \leq p$. The 0th moment partition function is independent of τ, i.e.,

$$g_\tau^{(0)}(\mathbf{n}) = g(\mathbf{n}), \tag{11}$$

which is the usual partition function in the literature. The computation of $g(\mathbf{n})$ is done by the established convolutional algorithm.[17] For moments $m = 1, 2$ the computation is done by specializing the recursions given in Ref. 20 to pseudo-networks.

It is noteworthy that for PANACEA we have devised a scheme for implementing the recursion in which only those $g_\tau^{(m)}(\mathbf{n})$ for which $\sum n_\sigma \leq 8$ are computed.

As each pseudo-network has t job classes there are $\binom{8+t}{t}$ such vectors $\mathbf{n}$, and $(1 + 2q)\binom{8+t}{t}$ computed values of $\{g_\tau^{(m)}(\mathbf{n})\}$.

In the computational phase of PANACEA the set of pseudo-networks is analyzed only once to compute all the expansion coeffcients $A^{(m)}_{\sigma,\tau,k}$. This implementation relies on $t(1 + 2pq)$ registers to be set up at the outset of the computational phase, one for each of the expansion coefficients. On completion of the computations pertaining to any pseudo-network, the computed data is used to update the relevant registers.

The implementation described above attempts to minimize not only arithmetical operations, but also virtual storage requirements.

3.4 Error bounds

A basic feature of PANACEA is that all computed quantities are accompanied by upper and lower bounds. This is made possible by the following theoretical result[19,20]:

$$\begin{aligned} I^{(m)}_{\sigma,\tau}(N) &\geq \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k \quad \text{if } t \text{ is even,} \\ &\leq \sum_{k=0}^{t-1} A^{(m)}_{\sigma,\tau,k}/N^k \quad \text{if } t \text{ is odd.} \end{aligned} \tag{12}$$

PANACEA computes the upper (lower) bound for the moments of the individual queue lengths given in (3) by using odd and even (even and odd) numbers of terms, respectively, in the asymptotic expansions for the integrals in the numerators and denominators of the expressions in (1). In certain cases (see Section 3.1) only one of the usual pair of bounds can be computed and in other, even rarer, cases no bounds can be computed.

IV. COMPUTATIONAL COMPLEXITY

We give a count of the total number of multiplications and the storage requirements for a problem solution on PANACEA. As multiplications are overwhelmingly more time-consuming than additions in double-precision floating-point operations, we omit a count of the latter.

4.1 Multiplications

Consider first the multiplications in the analysis of a particular pseudo-network. We have already noted in Section 3.4 that the number of vectors **n** that are valid arguments of $g^{(m)}_{\tau}(\cdot)$ is $\binom{8+t}{t}$. For each vector **n** as argument the convolutional algorithm for computing $g(\mathbf{n})$ requires tq multiplications, and to calculate $g^{(m)}_{\tau}(\mathbf{n})$, $m = 1, 2$ and all τ, there are $2tq$ additional multiplications. Thus,

$$\text{multiplications per pseudo-network} = (tq + 2tq)\binom{8+t}{t}. \tag{13}$$

Since only $\binom{p}{t}$ pseudo-networks are analyzed,

$$\text{multiplications total} = \binom{p}{t} 3tq \binom{8+t}{t} \tag{14}$$

$$\sim O(qp^t) \tag{15}$$

as q and p become large.

The remaining major component of the computations, the updatings of the registers (see Section 3.5), requires a total of less than 280 $\binom{p}{t}$ multiplications, which is usually negligibly small compared with the number in (14).

4.2 Storage

Again, consider first the requirement for a particular pseudo-network. The actual storage used in PANACEA is somewhat more than the number of computed quantities and is

$$(1 + 2q)10^4. \tag{16}$$

No further storage is required for the computations of all the pseudo-networks. There are also the $t(1 + 2pq)$ registers, a small number relative to (16).

4.3 Experimental results

We have conducted experiments on PANACEA to verify these results. Figure 4 presents PANACEA's response time for a particular network (see Fig. 4a) in which the number of terms in the asymptotic expansions $t = 4$. The results are for various values of the population **K** (Fig. 4b), the number of classes (Fig. 4c), and the number of active processing centers q (Fig. 4d). The broad features of the results on the response times are in agreement with the above complexity analysis: most importantly, **K** does not affect the response time and the dependencies on p and q are like p^4 and q.

V. NUMERICAL EXAMPLES

Several computational experiments have been conducted on PANACEA. These experiments were designed to highlight specific features of the package, explore its limitations, and to identify numerical stability issues. To date, we have not encountered difficulties with overflow or underflow in computing the asymptotic series. Hence, the rescaling of problem parameters that is often required to avoid numerical instability in the established algorithms is not at all an issue

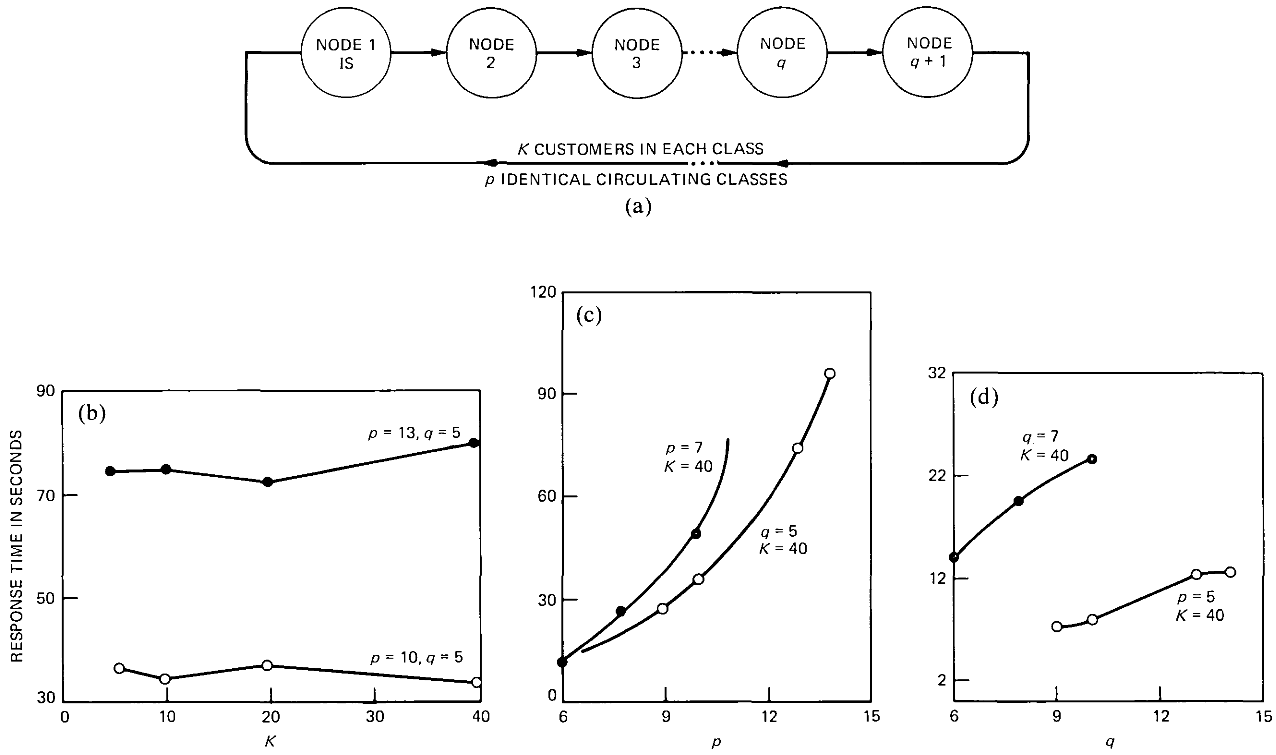


Fig. 4—(a) The network on which the experiments were conducted. Time complexity results for: (b) various values of the population K; (c) the number of classes; (d) the number of active processing centers, q.

here. The package has so far performed in a very stable and robust fashion in solving queueing network problems of widely differing sizes.

In all our computational experiments, the queueing network problems ranged in size from very small (one customer class with ten members in the class) to very large (17 customer classes with 17,000 total members in the customer classes). All problems were real-life problems arising in modeling large communication networks. The nature of the interaction of the performance analyst with the package involved solving a given problem, making quick parameter changes in the model, resolving the problem, and so on. This iterative mode of interaction implies the ability of the package to give fast response times, irrespective of the size of the problem. PANACEA was able to ensure a response time of less than 70 seconds for all problems by adaptively truncating the asymptotic series. (All problems were solved on VAX 11/780 with the *UNIX** operating system, Version 3.0, and lightly loaded <60%.) For reasons of brevity, we elaborate only on five numerical examples.

5.1 Example 1

The seven-node, nine-class queueing network shown in Fig. 1 (see the PML program shown in Fig. 2 for degrees of multiprogramming, service rates, etc.) was solved by PANACEA with a response time of about 20 seconds. Figure 5 shows the output displayed by PANACEA corresponding to the node DBP. Several features of Fig. 5 are noteworthy. First, PANACEA prints the number of terms computed in each asymptotic series. In solving this problem, three terms were computed. Second, the bounds printed for all quantities of interest are very "tight." For example, the DBP utilization for customer class 9 is 44.412201 and the upper bound on this utilization is 44.4305312 (an error of at most 0.05 percent). The standard deviations on queue lengths of all customer classes in the node DBP again show the "tight" nature of the bounds. Hence, with three terms in the series, a near-exact solution was obtained for the problem. This implies that for this particular problem, with a given set of parameters, the series converged rapidly.

5.2 Example 2

For some queueing network problems, one may not be able to achieve rapid convergence. This typically happens for "small" networks. Truncating the series too early may result in unacceptable errors. To demonstrate this phenomenon of slow convergence, we modeled the queueing network shown in Fig. 1 with one customer class

* Trademark of Bell Laboratories.

```
COMPUTATION PHASE BEGINS.
PACKAGE USES 3 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE
TIME ON THE VAX 11/780 HARDWARE
UTILIZATION STATISTICS ON PROCESSING NODES
UTILIZATION STATISTICS FOR NODE DBP
  CLASS      UTILIZATION         UPPER BOUND          LOWER BOUND
    1      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    2      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    3      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    4      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    5      1.68863533e - 01    1.68890352e - 01    1.68863533e - 01
    6      1.68879473e - 01    1.68906161e - 01    1.68879473e - 01
    7      9.95496911e - 04    9.95522116e - 04    9.95496911e - 04
    8      1.03167269e - 03    1.03169971e - 03    1.03167269e - 03
    9      4.44122010e - 01    4.44305312e - 01    4.44122010e - 01
MEAN QUEUE LENGTH STATISTICS
STATISTICS FOR NODE DBP
             MEAN QUEUE          UPPER BOUND          LOWER BOUND
  CLASS        LENGTH
    1      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    2      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    3      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    4      0.00000000e + 00    0.00000000e + 00    0.00000000e + 00
    5      7.61035551e - 01    7.61035551e - 01    7.37862183e - 01
    6      7.61104587e - 01    7.61104587e - 01    7.37942427e - 01
    7      4.50140929e - 03    4.50140929e - 03    4.37652979e - 03
    8      4.66486873e - 03    4.66486873e - 03    4.53537014e - 03
    9      1.99251497e + 00    1.99251497e + 00    1.92234003e + 00
STANDARD DEVIATION STATISTICS ON QUEUE LENGTH
STATISTICS FOR NODE DBP
  CLASS    STD. DEVIATION        UPPER BOUND          LOWER BOUND
    1      0.00000000e + 00           —                    —
    2      0.00000000e + 00           —                    —
    3      0.00000000e + 00           —                    —
    4      0.00000000e + 00           —                    —
    5      1.14408350e + 00    1.15916418e + 00    1.08306158e + 00
    6      1.14414970e + 00    1.15922376e + 00    1.08317002e + 00
    7      6.71791109e - 02    6.71873612e - 02    6.62227374e - 02
    8      6.83911390e - 02    6.83998675e - 02    6.74162160e - 02
    9      2.39306393e + 00    2.44979369e + 00    2.16185701e + 00
```

Fig. 5—Output of model example.

(this class required the services of DBP) with 20 terminals in the class. Table I shows the variation in the utilization of DBP as the number of terms in the asymptotic series is varied. Assuming that after accumulation of five terms, the series has converged to the exact solution, we see that truncating the series after the first term results in a 7-percent error, truncating the series after the second term results in a −4.4-percent error, and so on. Notice the oscillatory nature of convergence. After accumulation of four terms, we still have an error in the units position. Typically, this slow convergence occurs for "small" problems, and hence one can potentially compute more terms in the series without adversely increasing the time complexity. It is interesting to

Table I—Benefit of multiple terms

Number of Terms Used in Expansion	DBP Utilization	Error (%)
1	55.56	+7.0
2	49.63	−4.4
3	53.91	+3.8
4	50.68	−2.4
5	51.92	0

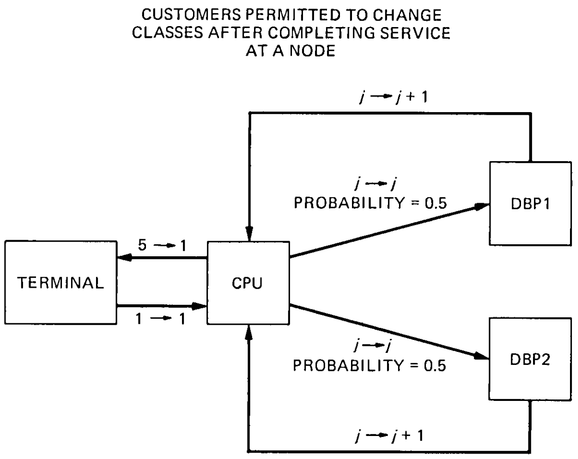
observe that slow convergence is an inherent property of the problem that cannot be cured by large choice of N. At first sight it might appear that simply by choosing N large enough, the convergence of the series may be speeded up. A closer examination of the terms of the asymptotic series reveals the fact that these terms are independent of N and depend only on the initial problem parameters.

5.3 Example 3

Table II shows results of comparing our package with a commercially available package, CADS.[25] CADS is marketed by Information Research Associates and it uses the convolutional algorithm to solve the queueing network. Again, the prototypical network shown in Fig. 1 was solved with one customer class and ten terminals in that class. Attempts at increasing the degree of multiprogramming or increasing the number of classes in the problem resulted in a breakdown of CADS. Hence, extensive comparisons of PANACEA and CADS on large problems could not be accomplished. Table II shows the results for a particular problem solved by both CADS and PANACEA and substantial agreement is exhibited. We have also validated PANACEA and CADS on many other "small" networks. On "small" networks with more than one class and about 10 terminals in all the classes, a significant improvement in the response time of PANACEA was observed, while near-identical results were printed by both PANACEA and CADS. This, of course, is corroborated by our time-complexity analysis (Section IV).

Table II—Comparison with CADS

	CADS		PANACEA	
Processor	Utilization (%)	Queue Length	Utilization (%)	Queue Length
DBP	26.1	0.338	26.08	0.3365
CPU1	4.35	0.0452	4.34	0.0452
FEP1	4.35	0.0452	4.34	0.0452



PROGRAM BEHAVIOR

ATOMIC OPERATION	CPU TIME IN MILLISECONDS	DBP TIME IN MILLISECONDS
OPEN FILE SYSTEM	200	80
OPEN FILE	50	40
READ	10	40
CLOSE	10	40

Fig. 6—Class-hopping model.

5.4 Example 4

This example illustrates the inter-class-transition feature of PANACEA. Version 1.1 of PANACEA permits "class hopping" between customer classes. Models with class-hopping features are the natural tools for analyzing precedence-constrained sequences of actions of a customer in a computer system. Figure 6 shows the central server model in which a terminal-generated request goes through a series of transactions in a strict sequence. Initially, the terminal request is processed by the CPU by executing a program. The germane aspects of the program behavior are captured in four atomic data base operations. Each data base operation set-up time in the CPU and the access time in either of the data base processors, denoted by DBP1 and DBP2, are shown in Fig. 6. When the terminal request is processed by the program, a series of data base transactions is generated to either of the DBPs with equal probability. The transactions, by their inherent nature, have to be executed in the following strict sequence: open file system, open file, read, and close. This precedence is enforced in the model by class hopping. The open-file-system transaction and the terminal request are both assigned a customer class label of 1. The other three atomic data base transactions are assigned class indices 2, 3, and 4, respectively. Class 5 is a "clean-up" class to gracefully

```
1 /*CLASS HOPPING EXAMPLE*/
2 OUTPUT CPU util + qlength + q2length;
3 /*ROUTING*/
4 CLASS TRANSITION              {1,2,3,4,5};
5 TER            :CPU           {⟨1⟩1.0};
6 CPU            :TER           {⟨5⟩{⟨1⟩1.0}};
  CPU            :DBP1,DBP2     {⟨1 to 4⟩.5};
  DBP1,DBP2      :CPU
                 {{⟨2⟩1.0},{⟨3⟩1.0},{⟨4⟩1.0},{⟨5⟩1.0}};
  /*SERVICE RATES*/
  CPU            {1000/200,1000/50,1000/10,1000/10,1000/5};
  DBP1,DBP2      {1000/80,⟨2 to 4⟩1000/40};
  TER            {⟨1⟩.05};
  CPU PS; TER IS;DBP1,DBP2 PS;
  DEGREE         {⟨1⟩40};
  END;
```

Fig. 7—PML program describing the class-hopping model.

terminate the terminal request. Now, all customers retain their class identity in all but three transitions. When a transition is made from DBP1 or DBP2 to CPU, a hop occurs from class j to class $j + 1$, as shown in Fig. 6 ($1 \leq j \leq 4$). Customer class 5 hops back to class 1 when transiting from the CPU to the terminal.

Figure 7 shows the PML program describing this model. A statement in line number 4 indicates class indices among which hops can take place. The routing statements appropriately denote the "from" class and the "to" class, if necessary.

Figure 8 shows a snapshot of the output produced by PANACEA. The upper and lower bounds are observed to be "tight" for all the quantities displayed in Fig. 8. The response time in solving this problem by PANACEA was almost instantaneous, once the out-of-core phase was completed. The out-of-core phase itself consumed about 60 seconds of elapsed time, on a lightly loaded VAX 11/780.

5.5 Example 5

We now consider the large communication network shown in Fig. 9, which consists of 23 processors, 17 classes and 1,000 terminals in each class. For reasons of brevity, we omit further description of this network. However, one can easily appreciate the explosion in state space of this queueing network and the CPU time that would be required to solve this problem by the previous methods. PANACEA was able to solve this problem almost instantaneously (response time was not perceived by the user). Figure 10 shows the output produced by PANACEA. Notice that only one term was computed for each asymptotic series. This implies, from the error analysis, that one of the bounds could not be computed for mean queue length and standard

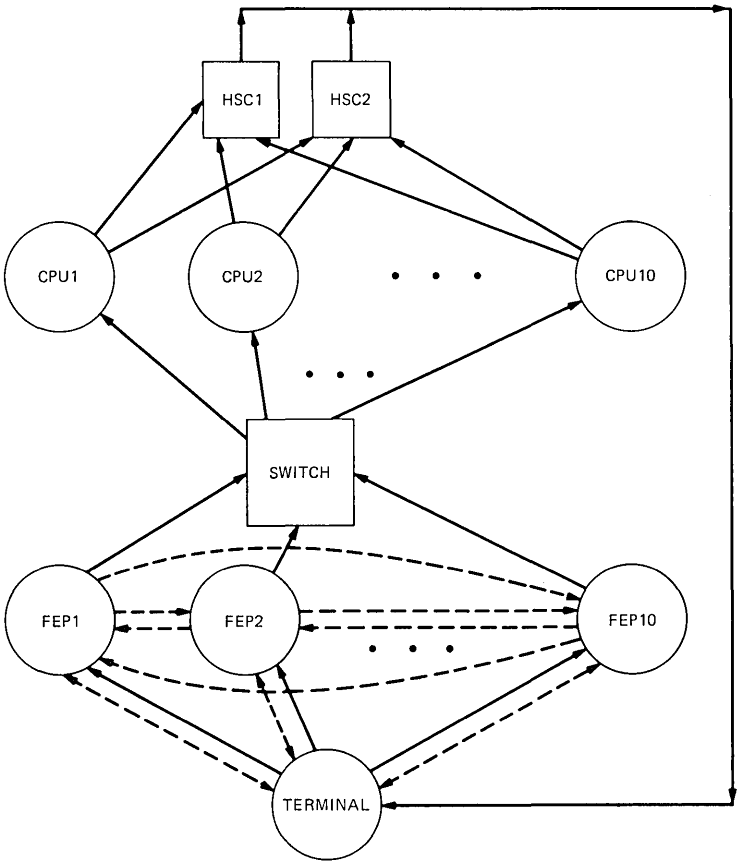
```
COMPUTATION PHASE BEGINS.

PACKAGE USES 4 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE TIME ON
THE VAX 11/780 HARDWARE

UTILIZATION STATISTICS ON PROCESSING NODES

UTILIZATION STATISTICS FOR NODE CPU

 CLASS    NODE UTILIZATION       UPPER BOUND          LOWER BOUND

   1       3.84754302e-01       3.84754302e-01       3.84336505e-01
   2       9.61885755e-02       9.61885755e-02       9.60841263e-02
   3       1.92377144e-02       1.92377144e-02       1.92168245e-02
   4       1.92377144e-02       1.92377144e-02       1.92168245e-02
   5       9.61885719e-03       9.61885719e-03       9.60841227e-03

MEAN QUEUE LENGTH STATISTICS

STATISTICS FOR NODE CPU

 CLASS  MEAN QUEUE LENGTH        UPPER BOUND          LOWER BOUND

   1       7.65705297e-01       7.93131456e-01       7.65705297e-01
   2       1.91426324e-01       1.98282864e-01       1.91426324e-01
   3       3.82852634e-02       3.96565713e-02       3.82852634e-02
   4       3.82852634e-02       3.96565713e-02       3.82852634e-02
   5       1.91426317e-02       1.98282857e-02       1.91426317e-02

STANDARD DEVIATION STATISTICS ON QUEUE LENGTH

STATISTICS FOR NODE CPU

 CLASS     STD. DEVIATION        UPPER BOUND          LOWER BOUND

   1       1.08703258e+00       1.19926103e+00       1.06718645e+00
   2       4.66285569e-01       4.88468353e-01       4.63411440e-01
   3       1.98305571e-01       2.03148498e-01       1.98035895e-01
   4       1.98305571e-01       2.03148498e-01       1.98035895e-01
   5       1.39293183e-01       1.42237373e-01       1.39197236e-01

END OF OUTPUT STATISTICS.

process terminated
*Q
Alice:▧
```

Fig. 8—Output of Example 4.

deviation statistics. Nevertheless, bounds on the utilization could be computed and they show acceptable accuracy. For example, the maximum percentage error occurs for customer class 14 in the node "hsc2" and it is less than 3 percent. Higher accuracy can be obtained only by computing the second element of the series at the expense of significant increase in response time.

VI. RECENT AND FUTURE DEVELOPMENTS

Recently, we have implemented Version 2.0 of PANACEA in which several additional features have been incorporated.



23 NODES, 17 CLASSES, 17,000 CUSTOMERS

Fig. 9—Large network of Example 5.

(*i*) Version 2.0 solves mixed queueing networks in which some job classes, or possibly no job classes, are closed, while the remaining classes are open.

(*ii*) The computational method described in this paper is supplemented by other established recursive techniques.[13-17] The package selects the latter for small networks and the method of asymptotic expansions for all other networks.

Future work will be directed to:

(*i*) Heavy usage asymptotics
(*ii*) Incorporation of load-dependent servers
(*iii*) Sparsity exploitation in solving pseudo-networks
(*iv*) ·Priority queue approximations.

```
COMPUTATION PHASE BEGINS.

PACKAGE USES 1 TERMS IN THE ASYMPTOTICS FOR GOOD RESPONSE TIME ON
THE VAX 11/780 HARDWARE

UTILIZATION STATISTICS ON PROCESSING NODES

UTILIZATION STATISTICS FOR NODE hsc2

 CLASS       UTILIZATION           UPPER BOUND          LOWER BOUND
   10     6.58941204e-02       6.60000106e-02       6.58941204e-02
   11     6.59778304e-03       6.60000119e-03       6.59778304e-03
   12     1.09938399e-02       1.10000014e-02       1.09938399e-02
   13     5.99891352e-03       6.00000113e-03       5.99891352e-03
   14     6.46839982e-01       6.60000114e-01       6.46839982e-01
   15     6.59868518e-03       6.60000119e-03       6.59868518e-03
   16     1.09963458e-02       1.10000014e-02       1.09963458e-02
   17     5.99891352e-03       6.00000113e-03       5.99891352e-03

MEAN QUEUE LENGTH STATISTICS

STATISTICS FOR NODE hsc2

 CLASS  MEAN QUEUE LENGTH      UPPER BOUND          LOWER BOUND
   10     3.87797861e-01       3.87797861e-01             —
   11     3.87797869e-02       3.87797869e-02             —
   12     6.46329747e-02       6.46329747e-02             —
   13     3.52543520e-02       3.52543520e-02             —
   14     3.87797866e+00       3.87797866e+00             —
   15     3.87797869e-02       3.87797869e-02             —
   16     6.46329747e-02       6.46329747e-02             —
   17     3.52543520e-02       3.52543520e-02             —

STANDARD DEVIATION STATISTICS ON QUEUE LENGTH

STATISTICS FOR NODE hsc2

 CLASS     STD. DEVIATION        UPPER BOUND          LOWER BOUND
   10     6.88753131e-01       7.90422721e-01             —
   11     1.99112072e-01       2.02853370e-01             —
   12     2.58917088e-01       2.66862288e-01             —
   13     1.89657360e-01       1.92906147e-01             —
   14     3.54066101e+00       5.25119023e+00             —
   15     1.99112072e-01       2.02853370e-01             —
   16     2.58917088e-01       2.66862288e-01             —
   17     1.89657360e-01       1.92906147e-01             —

END OF OUTPUT STATISTICS.
```

Fig. 10—Output of Example 5.

REFERENCES

1. F. P. Kelly, *Reversibility and Stochastic Networks*, New York: John Wiley, 1980, Chapter 3.
2. L. Kleinrock, *Queueing Systems. Vol II: Computer Applications*, New York: John Wiley, 1976, Chapter 4.

3. M. Schwartz, *Computer-Communication Network Design and Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1977, Chapter 11.
4. F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers," J. of the ACM, *22*, No. 2 (April 1975), pp. 248–60.
5. F. R. Moore, "Computational Model of a Closed Queueing Network with Exponential Servers," IBM Journal of Research and Development, *16*, No. 6 (November 1972), pp. 567–72.
6. J. Buzen, "Queueing Network Models of Multiprogramming," Ph.D. Thesis, Div. of Eng. and App. Sci., Harvard University, Cambridge, MA: 1971.
7. R. R. Muntz and J. Wong, "Asymptotic Properties of Closed Queueing Network Models," Proc. 8th Annual Princeton Conf. on Info. Sci. and Systems (March 1974), pp. 348–52.
8. D. P. Gaver and G. S. Shedler, "Approximate Models for Processor Utilization in Multiprogrammed Computer Systems," SIAM J. on Computing, *2* (September 1973), pp. 183–92.
9. M. Reiser, "A Queueing Network Analysis of Computer Communication Networks with Flow Control," IEEE Trans. Commun., *COM-27*, No. 8 (August 1979), pp. 1199–1209.
10. S. S. Lam and M. Reiser, "Congestion Control of Store-and-Forward Networks by Input Buffer Limits-An Analysis," IEEE Trans. Commun., *COM-27*, No. 1 (January 1979), pp. 127–34.
11. B. Pittel, "Closed Exponential Networks of Queues with Saturation: The Jackson-Type Stationary Distribution and Its Asymptotic Analysis," Math. of Oper. Res., *4*, No. 4 (November 1979), pp. 357–78.
12. E. Arthurs and B. W. Stuck, "A Theoretical Performance Analysis of a Markovian Switching Node," IEEE Trans. Commun., *COM-26*, No. 11 (November 1978), pp. 1779–84.
13. J. P. Buzen, "Computational Algorithms for Closed Queueing Networks with Exponential Servers," Commun. of the ACM, *16* (September 1973), pp. 527–31.
14. M. Reiser and H. Kobayashi, "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," IBM J. of Res. and Dev., *19*, No. 3 (May 1975), pp. 283–94.
15. M. Reiser and S. S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks," J. Assoc. Comput. Mach., *27* (April 1980), pp. 312–22.
16. C. H. Sauer and K. M. Chandy, "Computer Systems Performance Modeling, A Primer," Englewood Cliffs, NJ: Prentice Hall, 1981.
17. S. C. Bruell and G. Balbo, "Computational Algorithms for Closed Queueing Networks," New York: North-Holland, 1980.
18. J. McKenna, D. Mitra, and K. G. Ramakrishnan, "A Class of Closed Markovian Queueing Networks: Integral Representations, Asymptotic Expansions, Generalizations," B.S.T.J., *60*, No. 5 (May–June 1981), pp. 599–641.
19. J. McKenna and D. Mitra, "Integral Representations and Asymptotic Expansions for Closed Markovian Queueing Networks: Normal Usage," B.S.T.J., *61*, No. 5 (May–June 1982), pp. 661–83.
20. J. McKenna and D. Mitra, "Asymptotic Expansions and Integral Representation of Moments of Queue Lengths in Closed Markovian Networks," to be presented at the Int. Seminar on Modelling and Performance Evaluation Methodology, Paris, January 1983.
21. M. E. Lesk and E. Schmidt, unpublished work.
22. K. G. Ramakrishnan and D. Mitra, unpublished work.
23. G. M. Weinberg, *Psychology of Computer Programming*, New York: Van Nostrand, 1971.
24. Stephen C. Johnson, unpublished work.
25. "User's Manual for CADS," Austin, TX: Information Research Associates.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Special-Purpose Computer for Logic Simulation Using Distributed Processing

By Y. H. LEVENDEL, P. R. MENON, and S. H. PATEL*

(Manuscript received March 20, 1982)

This paper presents the architecture of a special-purpose computer for logic simulation using distributed processing. The architecture is based on the utilization of inexpensive microprocessors interconnected by a communication structure. The communication structure is cross-point based for simple evaluations and time-shared parallel bus based for functional evaluations. Analysis is carried out to show that the performance of the proposed simulator is better by over two orders of magnitude than traditional logic simulation carried out on a general-purpose computer. Also, the power of the simulator is proportional to the number of slave processors over a certain range.

I. INTRODUCTION

The logic circuit simulator is an important component of a CAD system. It is used to predict logic circuit operation and performance under normal and faulty conditions. The application of the logic circuit simulator can be divided into two major areas: verification of new logic hardware designs and fault analysis of these designs.

As an evaluation tool, it can be used to verify the logical correctness of new hardware designs. Other information that can be obtained using a logic simulator include timing and signal propagation characteristics, and race and oscillatory circuit conditions. If the results of simulation indicate an unsatisfactory design, i.e., the circuit does not perform as expected, then changes can be made to the design. The design can be reevaluated using the simulator. After a number of iterations, a satisfactory design ready for committing to hardware should result. A logic circuit simulator used as above is known as a true value simulator.

* This paper is based on material to be submitted by S. H. Patel in partial fulfillment of the requirements for the Ph.D. in Electrical Engineering at the Illinois Institute of Technology.

Note that the results of true value simulation can be used for diagnosing faulty equipment using the guided probe technique.[1]

Most logic circuit simulators also have fault-simulation capability. This capability can be used to determine fault coverage by a proposed test sequence, production of a fault dictionary, evaluation of test-point effectiveness, and evaluation of self-checking circuitry. The behavior of a circuit under fault conditions can also be investigated. Fault analysis can be used to investigate initialization and fault-induced races, and to perform timing analysis under specific fault conditions. For fail-safe circuitry, selected faults can be inserted in the circuit and the effect of these observed on the outputs by simulation. If a forbidden output is obtained under some fault, then the circuit design must be changed.

Figure 1 shows the general environment of a logic circuit simulator. The circuit to be analyzed is modeled using a circuit-description language. This language describes the connectivity and behavior of the circuit. The modeling information typically includes element type (gate or functional), associated delays, and interconnection data. Once the data structure of the model is set up in the logic circuit simulator, simulated inputs are applied either dynamically (at prescribed times) or statically (after the circuit is stabilized). Fault simulation is performed using one of several methods: one fault at a time, parallel, concurrent, or deductive.[2-4] The simulated output is recorded either in a plot form (true value) or tabular form (true value and fault simulation).

Currently, most digital circuits are simulated on large general-purpose computers. This method of simulation is complex and expensive to operate and maintain.[5] There is a need for more sophisticated and cost-effective simulators as we get into the VLSI era. Very large simulation time and costs will result when dealing with circuits of VLSI complexity (more than 100,000 gates on a single chip).

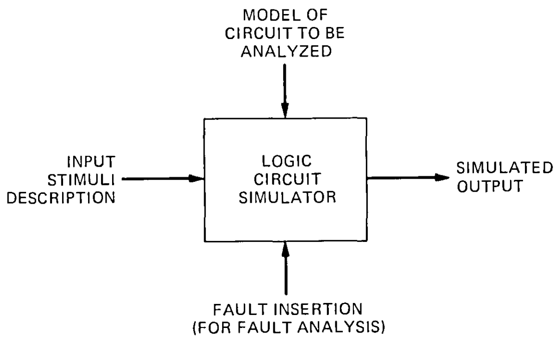

MODEL OF CIRCUIT TO BE ANALYZED
INPUT STIMULI DESCRIPTION
LOGIC CIRCUIT SIMULATOR
SIMULATED OUTPUT
FAULT INSERTION (FOR FAULT ANALYSIS)

Fig. 1—Operating environment of a logic circuit simulator.

II. SPECIAL-PURPOSE SIMULATION HARDWARE

Existing logic circuit simulators are implemented in software that is executed on a general-purpose computer. To date, a large amount of work has been invested in optimizing this software. For further improvements in the performance of the logic circuit simulator, the hardware on which the simulation software executes must be optimized. With the advent of low-cost microcomputers, the development of special-purpose logic simulation hardware becomes attractive. Possible benefits are higher speeds, lower costs, and greater flexibility (for example, better integration in a test station).

Recently there has been some interest in developing special-purpose logic simulators. Barto and Szygenda have developed special-purpose simulation hardware based on distributed processing.[5] More recently, Abramovici et al. have presented a special-purpose architecture based on pipelining and concurrency.[6] Both approaches use dedicated processors for performing specific tasks. A special-purpose logic simulation machine using parallel processing (the Yorktown Simulation Engine) has been built by IBM.[7-9]

Concurrency allows simultaneous processing of several parallel events leading to a reduction in overall processing time. There are at least two types of concurrencies present in the simulation of logic circuits. One type of concurrency occurs in the simulation algorithm and the other in the actual simulated hardware.

The first type of concurrency can be called *algorithm concurrency*. In logic circuit simulation a number of operations have to be performed during a simulated time interval. Simulated time consists of discrete points in time (approximated to the nearest integer) at which changes in logic values on signal lines can occur. A simulated time interval is the time between two such consecutive discrete points. A simulated time interval is also sometimes referred to as a *time frame*. A *simulation cycle time* is the time required to carry out the processing during a simulated time interval. Typical operations carried out during a simulation cycle include determining current events, updating values at source, determining fanout, updating values at fanout, evaluating elements, and scheduling resulting events. An *event* is a change in logic value on a signal line. Scheduling an event is marking it to occur at some time in the future. Consider several elements being evaluated and several resulting events being scheduled during a simulation cycle. In traditional simulation, the elements are evaluated and the resulting events scheduled sequentially. No two operations are performed simultaneously. One can take advantage of the inherent concurrency by noting that after an element has been evaluated and while the resulting event is being scheduled, the evaluation of another element can be concurrently started. An average number of 80 such concurrent events were observed per simulated time interval during the simulation of a 4000-gate circuit.[6] This concurrency appears in the simulation algorithm. The architecture proposed by Abramovici et al. takes advantage of this concurrency to some extent. The main ingredient of this solution is *functional partitioning* of the tasks to several microprocessors.

The concurrency can also be viewed from the point of view of the logic circuit. Concurrent events occur during a simulation cycle because of the way electrical signals propagate in the logic circuit. Several elements may be activated at the same time because signal propagation occurs simultaneously along several paths in the actual hardware. If the elements that become active at the same time are processed by different processors simultaneously, then the overall simulation time will be reduced. This type of concurrency can be called *logic circuit concurrency.* Its main ingredient is *distributed processing* among several processors, all of which are executing the same algorithm.

This paper describes the architecture of a special-purpose logic simulation machine designed to take advantage of the parallelism caused by concurrent activity of signals in a circuit. The system is essentially a processing network based on an interconnection of low-cost microcomputers. The circuit to be simulated is partitioned into subcircuits and each subcircuit is simulated in a separate microcomputer. Thus, several microcomputers can be simultaneously simulating several elements activated by parallel signals. This simulator is different from those proposed in the literature (see Refs. 5 and 6) in that the multiple processors do not perform dedicated tasks. Also, the modularity of the simulator proposed in this paper allows easy increase of computational power.

III. MULTIPROCESSOR OPERATION ENVIRONMENT

The operation environment of the multiprocessor digital logic simulator is shown in Fig. 2. The general-purpose computer acts as a preprocessor at the beginning of simulation and as a postprocessor at the end of simulation.

At the beginning of simulation, the circuit to be simulated is modeled on the general-purpose computer. The data structure is then loaded into the multiprocessor simulator. The loading problem is not discussed in this paper. After setting up the environment for the multiprocessor simulator, the general-purpose computer requests the simulator to start.

The simulation is carried out in the multiprocessor simulator. The simulator can be programmed to output intermediate results automatically to the general-purpose computer. The simulator can also be interrupted by the general-purpose computer for intermediate results. The user can ask for information about a simulation run while it is in

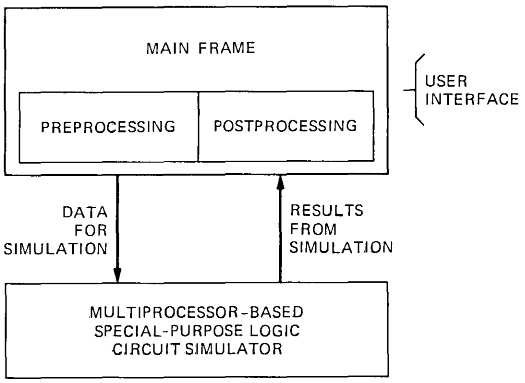


Fig. 2—Preprocessing and postprocessing for the multiprocessor-based special-purpose logic simulator.

progress (e.g., the status of a variable) and make certain run-time decisions such as continue simulation, apply extra input vectors, or stop. At the end of circuit simulation, the final simulation results and any other user-requested information is sent to the general-purpose computer. User-requested information typically includes output values of elements (monitored points) at specific simulated times or under some other specified conditions. The general-purpose computer formats this information for suitable presentation to the user.

IV. ARCHITECTURE DESCRIPTION

The multiprocessor simulator consists of processors p_1 through p_n. The circuit to be simulated is partitioned into blocks a_1 through a_n. The signal connections between two blocks a_i and a_j are designated as b_{ij}. Each block a_x is then mapped into processor p_x as a subcircuit c_x. Figure 3 shows two blocks a_i and a_j mapped to processors p_i and p_j, respectively, as subcircuits c_i and c_j. The blocks are not necessarily clusters. That is, elements in a block can be from disjoint portions of the circuit. The signal connections b_{ij} between blocks a_i and a_j are mapped in a data path d_{ij} between processors p_i and p_j.

During simulation the subcircuits c_i and c_j are simulated independently. Different subcircuits become active as signal values proceed from the primary inputs to primary outputs. As simulation progresses, data will have to be carried between subcircuits c_i and c_j as the logic values on the signal connections between the two subcircuits change. This data is transported across the data path d_{ij}. Typical data sent across the data path consists of changed logic values.

The architecture of the multiprocessor simulator proposed in this paper is shown in Fig. 4. The simulator consists of a communication

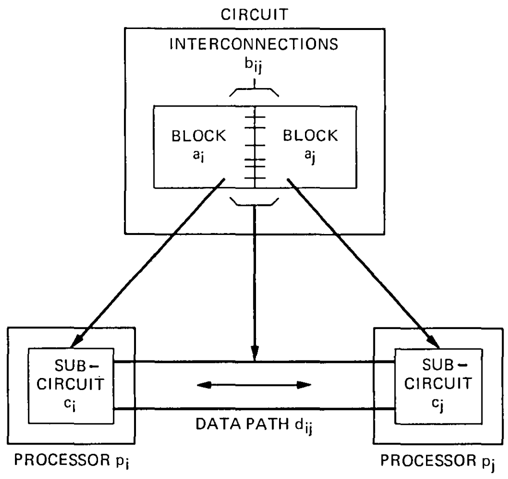


Fig. 3—Mapping of partitions a_i and a_j to processors p_i and p_j.

structure (communication medium and its associated control) connected to a master, several simple evaluators for simulating gate level blocks, and several functional evaluators for simulating functional blocks. A cross-point matrix is used to interconnect the master and the simple evaluators. The functional evaluators are connected to the cross-point matrix through a bus interface unit and a parallel bus. It is shown in Section VI that the speed of a cross-point matrix is required for transferring data between the simple evaluators. A parallel bus provides sufficient speed for functional evaluators. Note that if only

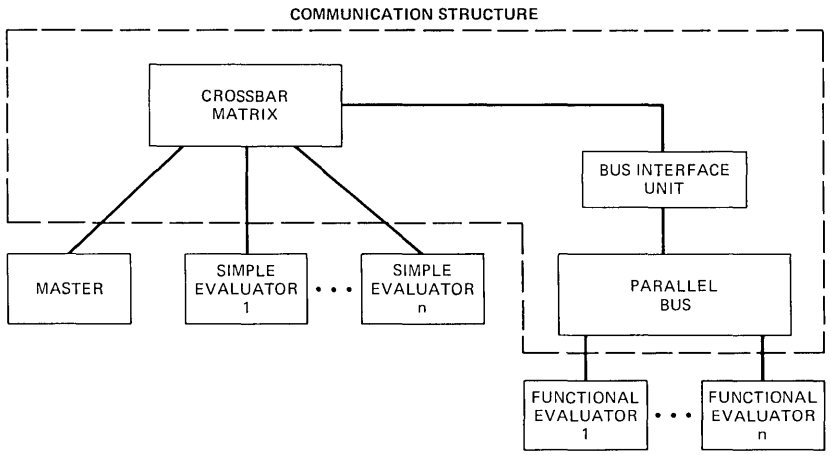


Fig. 4—Architecture of proposed multiprocessor simulator.

simple evaluations are to be carried out, then the bus interface unit, the parallel bus, and the functional evaluators can be removed. If only functional evaluations are to be performed, then the bus interface unit, the cross-point matrix, and the simple evaluators can be removed.

V. MULTIPROCESSOR IMPLEMENTATION

The configuration of the multiprocessor-based digital logic simulator is shown in Fig. 5. The simulator consists of one master and a multiplicity of slaves interconnected by a communication structure (communication medium and its associated control). The implementation of the simulator allows the use of either a shared or a dedicated communication structure. The master has local memory for its use. Each slave consists of a processing unit (PU) with its associated memory. The master and the slaves also each have two FIFO buffers and two data sequencers for interfacing to the communication structure (see Sections 5.1 and 5.2).

The processors p_i and p_j shown in Fig. 3 correspond to the slaves s_i and s_j in the multiprocessor simulator. The subcircuits c_i and c_j reside in the memories of the slaves s_i and s_j. The interconnections b_{ij} between blocks a_i and a_j are mapped into the data path d_{ij} that constitutes part of the communication structure.

At the beginning of each simulation cycle the master sends primary input values (if any) to the appropriate slaves using the communication structure. The master then issues a start signal to the slaves. This signal informs the slaves to start processing for the next simulation cycle. During the processing of a simulation cycle a slave unit may generate data for the other slaves or the master. The data is sent to the destination slave or the master using the communication structure. Data transferred between the slaves consists of scheduled events for the next time frame. A *scheduled event* is a change in logic value on a signal line scheduled to occur at some time in the future. Only data for the subsequent time interval is transferred between the slaves in order to reduce the amount of information sent over the communication structure, and thus the communication overhead. The scheduled time does not have to be sent. Data transferred from the slaves to the master consist of primary output values and user-requested information.

Each slave informs the master when it has finished processing and transferring data for the current simulation cycle. When all slaves have informed the master about their completion of processing for the current simulated time interval, and also the master has finished transferring any primary input values scheduled for the next simulated time interval to the slaves, the master issues a start signal to the slaves for the next simulation cycle.

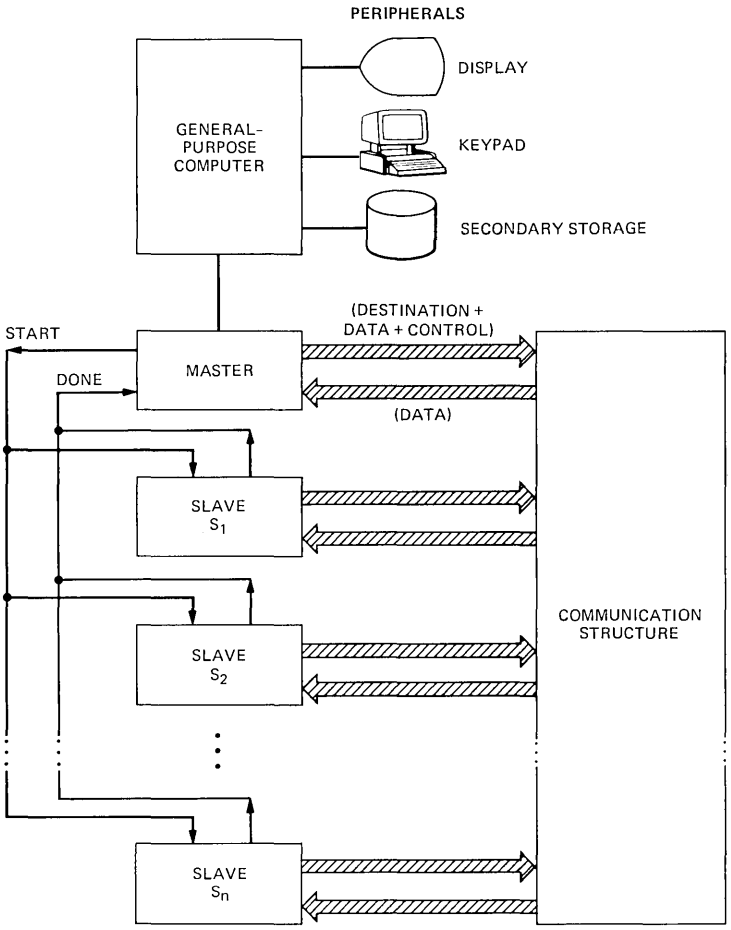


Fig. 5—Implementation of a multiprocessor-based digital logic simulator.

The different sections of the multiprocessor are discussed in greater detail below.

5.1 Slave unit

The slave unit configuration is shown in Fig. 6. The PU is a general-purpose 16-bit microprocessor. The input and output data sequencers can be either specially designed logic circuits or commercially available single-chip microcomputers. The FIFO buffers are commercially available devices.

The slave unit PUs perform the actual element/function evaluation and event scheduling. As noted previously, the partitioning of the logic circuit to be simulated is done on a general-purpose computer. Each

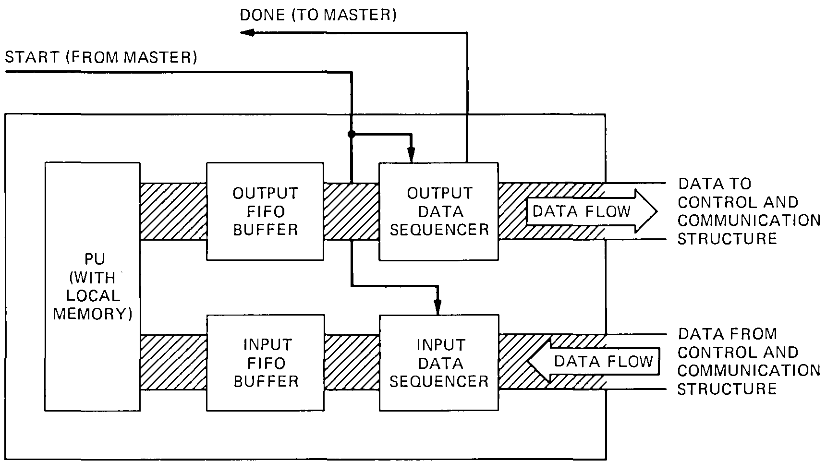


Fig. 6—Slave unit configuration.

PU contains a block of the logic circuit to be simulated. The simulation algorithm resides in the PU memory; all slaves contain identical algorithms.

Each slave uses an Output Data Sequencer (ODS) to transfer data out and an Input Data Sequencer (IDS) to receive data from the other slaves or the master via the communication structure. In a slave unit, the PU and the data sequencers are isolated from each other by means of FIFO buffers. Thus, the slaves can send and receive data independently of whether the PU is active or not.

The PU stores any data it has for other PUs or the master in the Output FIFO Buffer (OFB). The ODS makes a request for the communication structure if there is any data to transfer from the OFB. The ODS of the slave, if granted the use of the communication structure, takes data (scheduled values for slaves and primary output values and user-requested information for the master) from the OFB and sends it over the communication structure to other slaves or the master. The data is received by the IDS of the destination slave or the master (described in Section 5.2). Any data received by an IDS is put in its Input FIFO Buffer (IFB). End of Data (EOD) flags are used to separate data streams since a PU can be writing new data to the OFB before its ODS has finished transferring current data and similarly, an IDS can be receiving new data in the IFB before its PU has finished reading current data.

There are two signal lines between a slave unit and the master. The master signals the slaves using a **START** signal and the slaves signal the master using the **DONE** line. The **DONE** line will become active when all the slaves have finished processing.

The **START** line from the master initiates the processing for the next PU processing cycle. This signal causes the IDS to load an EOD flag in the IFB of all slaves. At the beginning of each simulation cycle, the PU monitors the IFB for data from other slaves and the master for the current simulation cycle. The EOD flag marks the end of data from other slaves and the master for the current simulation cycle. When the PU reads this flag, it starts evaluation for the current simulation cycle. The **START** signal from the master also informs the slave ODS to start sending out any data to be used during the next simulated time cycle from the OFB.

At end of the simulation cycle the PU loads an EOD flag in the OFB and starts preprocessing for the next simulation cycle. When the ODS encounters the EOD flag in the OFB, it has finished transferring data for the current simulation cycle. The ODS informs the master using the **DONE** line.

5.1.1 PU operation

The following data tables are used by each PU for its operation (see also Fig. 7):

(*i*) *Circuit Description Table.* This table contains interconnection data for the subcircuit. For each element it contains the value, type, delay, input status word pointer and corresponding status fields, internal fanout list pointer and corresponding fanout lists, and external fanout list pointer and corresponding fanout lists. The input status word pointer and the corresponding status fields give the signal values on the fanin lines. The internal fanout list pointer and corresponding fanout lists give the fanout which remain in the subcircuit. The external fanout list pointer and corresponding fanout lists give fanout which go to subcircuits located in other slaves. An element may have only internal fanout, only external fanout, or both internal and external fanout. Note that storing the external fanout takes up more space than storing an internal fanout since both the destination processor address and element index have to be stored for the external fanout.

(*ii*) *Activity List.* This list is used to keep track of active elements during a simulation time interval. These elements are to be evaluated.

(*iii*) *Timing Wheel.* This data area contains the events that are scheduled in the future. A large amount of work has been done in this area.[10,11]

The PU operation can be described in terms of two essentially concurrent processes, namely the simulation cycle (execution of simulation algorithm) and the communication cycle (communication of events). During one simulation cycle the following operations occur in the given order:

1. Update line values from current list L_t of timing wheel.

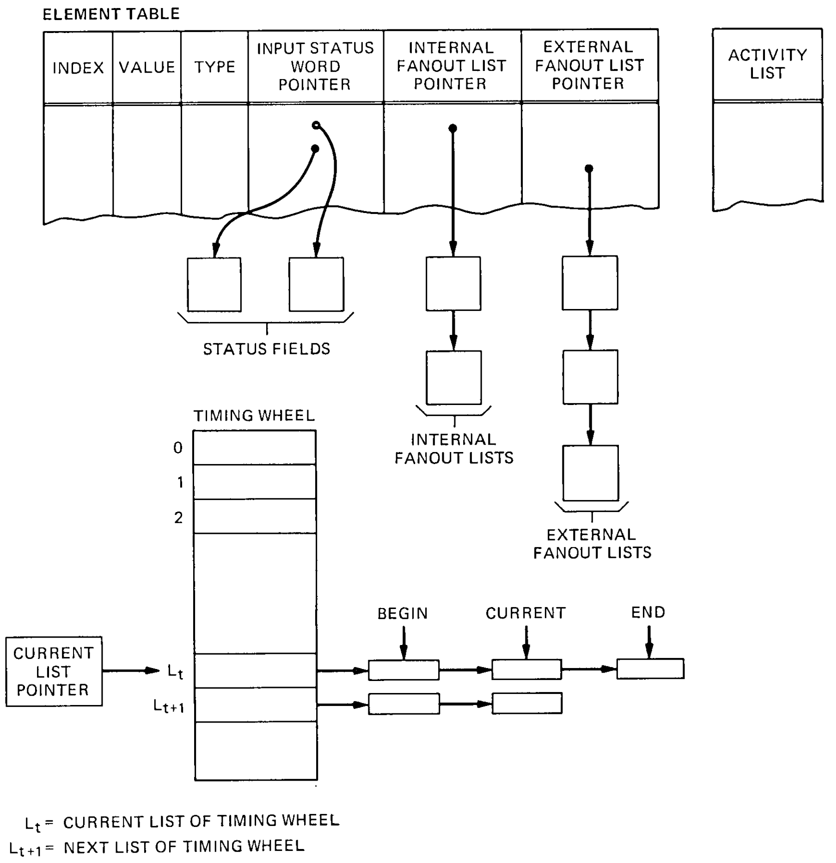


Fig. 7—Data tables for PU operation.

2. Find fanout and put active elements in activity list.

3. From next list L_{t+1} of timing wheel, for each entry that is an external fanout node, store scheduled event in OFB. Remove entry from timing wheel if it does not have any internal fanout also.

4. Update line values from IFB until EOD flag is received. The EOD flag signifies end of data present in the IFB for the current simulation cycle. (The EOD flag is loaded by the IDS when it receives a **START** signal from the master.)

Note: Any user requests received are stored for later processing.

5. Find fanout and put active elements in activity list.

6. Evaluate elements in activity list.

7. For active elements whose output changes, if delay of element is one and it is an external output node, schedule change in OFB.

8. If test in step 7 fails, schedule change on timing wheel.

9. Gather any user-requested information and store it in OFB.

10. Store EOD flag in OFB.

11. Increment current list pointer of timing wheel to the list of next time.

From steps 7 and 8 it can be seen that events on an external fanout are scheduled on the timing wheel of the processor in which the event occurred except in the case where the external fanout is going to be active in the next time interval. In this way the scheduled time does not have to be transmitted with the scheduled event, thus saving communication overhead.

A communication cycle is the period in between two **START** commands issued from the master. This cycle is phased with respect to the simulation cycle, as shown in Fig. 8. Note that the end of a communication cycle is an appropriate point for the multiprocessor simulator to stop for processing any interrupt requests from the general-purpose computer or sending out intermediate results. The master can issue the start for the next simulation cycle by sending a **START** command to the slave data sequencers after it satisfies all requests.

All the slave unit PUs contain the same software. Note that this algorithm is similar to the one used in traditional logic simulators except that the operations are sequenced differently.

5.1.2 Operation of slave data sequencers

The function of the data sequencers (IDS and ODS) is to transfer data from the OFB to the communication structure and from the communication structure to the IFB.

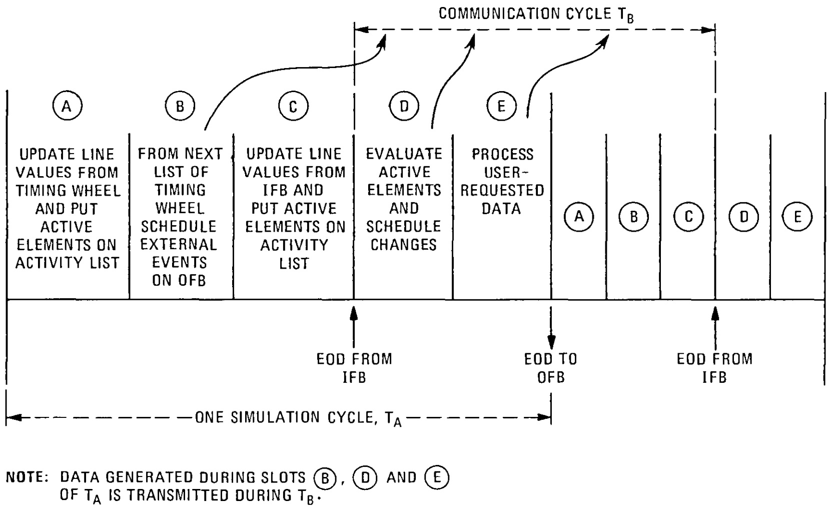

COMMUNICATION CYCLE T_B
A
B
C
D
E
UPDATE LINE VALUES FROM TIMING WHEEL AND PUT ACTIVE ELEMENTS ON ACTIVITY LIST
FROM NEXT LIST OF TIMING WHEEL SCHEDULE EXTERNAL EVENTS ON OFB
UPDATE LINE VALUES FROM IFB AND PUT ACTIVE ELEMENTS ON ACTIVITY LIST
EVALUATE ACTIVE ELEMENTS AND SCHEDULE CHANGES
PROCESS USER-REQUESTED DATA
A
B
C
D
E
EOD FROM IFB
EOD TO OFB
EOD FROM IFB
ONE SIMULATION CYCLE, T_A
NOTE: DATA GENERATED DURING SLOTS B, D AND E OF T_A IS TRANSMITTED DURING T_B.

Fig. 8—Relationship between a simulation cycle and the communications cycle.

The IDS and the ODS have some local memory. The IDS uses this memory to store any data it receives from the outside in case the IFB overflows. The ODS requires this memory only in certain circumstances. A case for its use is given Section 5.3.2.

The communication protocol between the data sequencers and the communication structure depends on the type of communication structure. This is discussed in detail in Section 5.3.

5.2 Master processor

The master processor is the interface between the general-purpose computer and the simulator. Its main functions are to keep track of simulated time, keep the slaves in synchronism, supply the slaves with primary input values, and gather the primary output values from the slaves. It also stores any user-requested monitored point values sent to it by the slaves.

The configuration of the master is similar to that of a slave unit and is shown in Fig. 9. It consists of a central processing unit (CPU) with some local memory, an input FIFO buffer (IFB), an output FIFO buffer (OFB), an input data sequencer (IDS), and an output data sequencer (ODS). The master is connected to the slave units through the communication structure. The master initiates processing for the next simulation cycle by issuing a **START** command on its signal line. When the slaves finish processing for the current simulated time interval, they inform the master through the **DONE** signal.

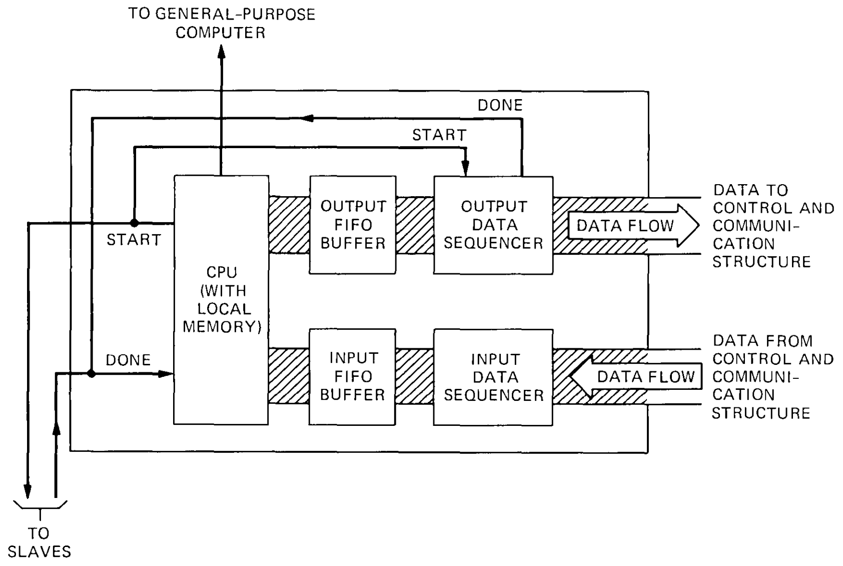


Fig. 9—Master configuration.

The master sends and receives data to and from the slaves using the communication structure. The interface between the master and the communication structure is similar to that between the slaves and the communication structure. The master stores any data (primary input values or user requests for monitored point values) it has for the slaves for the next simulation cycle in its OFB. The **START** signal which goes to the slaves also informs the ODS of the master to start transferring out data. The ODS encounters an EOD flag in the OFB when it has transferred all the data from the OFB. The ODS informs the master that it has finished sending out data by setting the **DONE** line. The IDS receives data from the slaves and puts it in the IFB.

Figure 10 shows the master, slave unit PU, slave unit ODS, and slave unit IDS operations during a simulation cycle.

5.3 Communication structure

The communication structure is used as a medium for transferring data between the slaves and between the slaves and the master. Either a shared or a dedicated structure can be used for the multiprocessor simulator. Two types of communication structures will be considered here, namely the time-shared parallel bus and the cross-point matrix. Each case is treated separately below. The criteria for selecting the type of communication structure are given in Sections VI and VII.

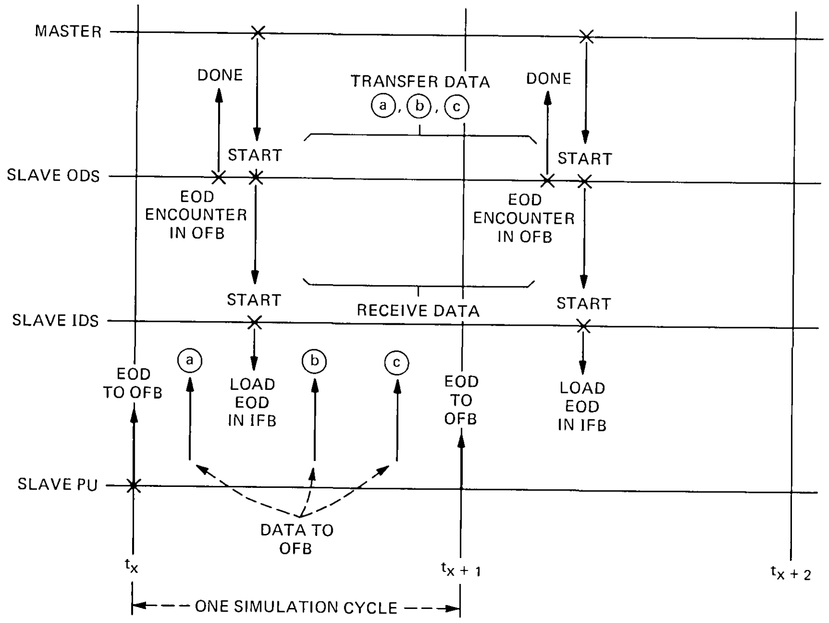


Fig. 10—Master and slave unit operations during a simulated time interval.

5.3.1 *Time-shared parallel bus*

The interface between the data sequencers and a time-shared parallel bus based communication structure is shown Fig. 11. When the ODS has some data to send, it sets the Request To Send (RTS) line high. The bus control grants it the use of the communication medium by sending a pulse on the Bus Grant line. The ODS sends out all the data present in its OFB. The data received by the IDS of the destination unit is put in its IFB. The ODS then sets the RTS line low. This releases the bus, which is then granted by the bus control to another requesting slave or the master. All units have equal priority. The ODS will set the RTS line high again if it gets more data to transfer in the OFB.

The data sent out to a slave unit from another slave unit or the master consists of a scheduled event for the next simulation cycle. The data sent to the master consists of the address of the sending slave, element number (primary output or monitored point) and element value. A separate line Request to Send to Master (RTSM) is used to address the master. When the destination is the master, the address lines from the ODS contain the sending slave unit address. This address together with the element number and element value is stored in the master IFB by the master IDS.

5.3.2 *Cross-point matrix*

The interface between the data sequencers and a communication structure based on a cross-point matrix is shown in Fig. 12. When the ODS has some data to send, it puts the address of the destination unit on the address bus and makes a request to transfer data by sending a pulse on the RTS line. If the destination is not busy, the control for

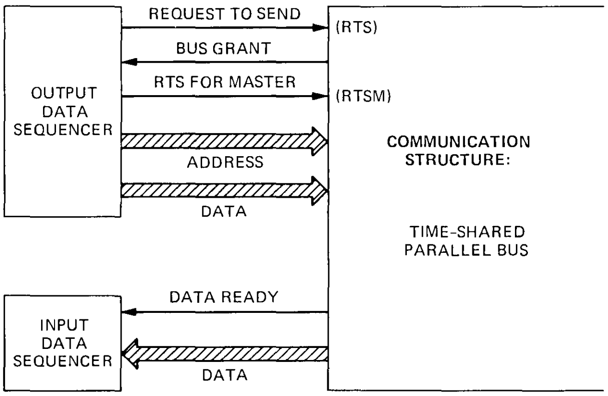


Fig. 11—Interface between the data sequencers and a time-shared parallel bus.

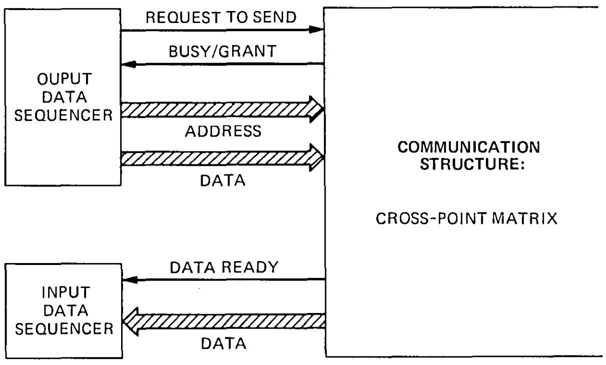


Fig. 12—Interface between the data sequencers and a cross-point matrix.

the matrix grants the transfer request. The ODS sends out data serially over the data line. The data received by the IDS of the destination unit is put in its IFB. The Data Ready line connected to the IDS is used to show the presence of data. It might happen that the destination is busy when an ODS makes an RTS to the cross-point matrix. In this case, the ODS gets a busy signal from the control of the cross-point matrix. In response to the busy signal, the ODS stores away the data that was to be transferred in its local memory and makes an RTS for the next set of data to be transferred from the IFB. A retry to send the blocked data is made later.

As indicated in Section 5.3.1, the master requires the address of the sending slave unit. A slave unit ODS will recognize a request to transfer data to the master and it will transmit its slave unit address together with element number and value.

The interface discussed above will apply for a nonblocking switching network also. However, this network will not be discussed here.

VI. ARCHITECTURE EVALUATION

In this section, various performance functions are derived for the multiprocessor architecture and compared with those for the traditional logic simulator implemented on a general-purpose computer. The requirements for a circuit-partitioning algorithm are considered first. Based on these requirements, expressions and values for processing and communication times are derived next. Comparisons in terms of evaluations per second are then made between the multiprocessor simulator and the traditional logic simulator implemented on a general-purpose computer.

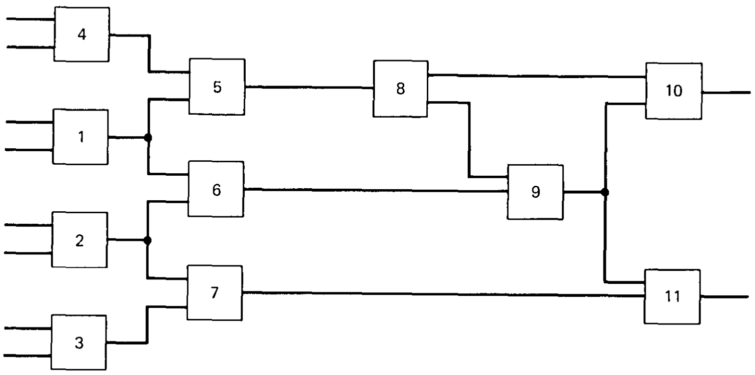


Fig. 13—A logic circuit.

6.1 Logic circuit partitioning

The logic circuit to be simulated is partitioned into a number of subcircuits. Each subcircuit can essentially be regarded as an independent circuit. During simulation, data is passed between different subcircuits as signal values propagate from the primary inputs to the circuit outputs. The subcircuits will be stored in the appropriate memories of the slave unit PUs. (The element numbers in the original circuit are translated to a slave address and an index number.) Partitioning is a key to the operation and performance of the multiprocessor digital logic simulator. Partitioning must maximize multiprocessing while limiting communication.

A partitioning algorithm is required to partition a logic circuit into subcircuits. The partitioning algorithm must produce subcircuits such that during logic circuit simulation, the number of simultaneously active subcircuits (processors) is maximum and the number of simultaneously active elements in each subcircuit (processor) is minimum, while keeping the communication from being a bottleneck. Obviously, minimization of interprocessor communication and the proper choice of communication structure are necessary to avoid this bottleneck (see Sections VII and VIII). The fact that signals may propagate in parallel indicates that partitioning should be done along the depth of the circuit rather than the breadth of the circuit since this will tend to place concurrent activities in different blocks. One approach is to start with a primary input and trace a path towards a primary output forming an element string. An element string is a single fanout path. For example, elements (1, 6, 9, 11) and (5, 8, 10) in Fig. 13 constitute two element strings. Since two elements in a string will not be normally active simultaneously, the whole string can be put in one subcircuit. Each subcircuit corresponds to a block described in Section IV. Note

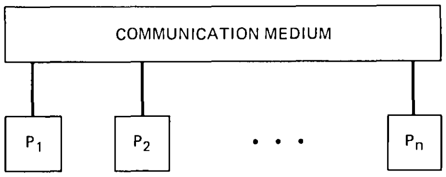


Fig. 14—Multiprocessor simulator architecture.

that if there is a fanout from any element to a zero delay element, then the two elements must be in the same block. For a multiprocessor system with n processors, a logic circuit must be partitioned into at most n blocks. A partitioning algorithm based on this approach is given in Appendix A.

6.2 Processing and communication

Consider the architecture of a multiprocessor simulator shown in Fig. 14. Processors p_1 through p_n are connected by a communication structure. The logic circuit to be simulated is partitioned into n blocks using the partitioning algorithm described in Appendix A. Each block is then assigned to a processor. During simulation, the communication structure will be used to transfer data between the processors as the interconnections between different blocks become active.

During a simulation cycle each of the processors will be evaluating active elements and scheduling events. Concurrently, data will be flowing across the communication structure as events in one processor activate elements in another processor. We define t_p as the average processing time per processor during a simulation cycle. Time t_p represents the processing of all active elements and scheduling of resulting events in one processor during a simulation cycle. Define t_c as the total communication time during one simulation cycle. This value represents the amount of time the communication structure is busy (i.e., the time taken to service all requests to transfer data from all processors) during a simulation cycle. Since the operations during t_p and t_c occur concurrently, the length of the simulation cycle and, hence, the number of evaluations per second will be determined by the greater of t_p and t_c.

Expressions and estimates for t_p and t_c for an optimum architecture will be derived in the next two sections. The value of t_c will be derived for two cases, namely, a communication structure based on a time-shared parallel bus and a cross-point matrix.

6.3 Processing time t_p

Let N be the average number of active elements per simulation cycle and n be the number of processors in the multiprocessor simulator. Then N/n will be the average number of active elements per processor during a simulation cycle for the ideal case. There will be some extra active elements in a processor during a simulation cycle due to nonideal partitioning. If k is the average unbalance factor, then the number of active elements is kN/n. Let a be the time required to process one active element, then:

$$t_p = (N/n)\,ka, \quad 1 < n < N;\ k = 1 \text{ for } n = 1.$$

This expression gives the average processing time per processor during one simulation cycle.

The value of the processing time a is estimated next. Traditional logic simulators can perform about 30,000 simple evaluations per second (e.g., IBM 370/168). This represents 33 μs per element evaluation. The simulation algorithms for the multiprocessor simulator and the traditional logic simulator are somewhat similar. The element evaluation time for the multiprocessor simulator can be written as $a = 33u_1$ μs. The factor u_1 represents a slowdown factor due to the difference in speed between a microprocessor and a general-purpose computer. For an Intel* 8086 16-bit microprocessor the slowdown is about 5.5 over an IBM 370/168.[12] The element evaluation time for an Intel 8086 becomes 181.5 μs. The operation of the microprocessor can be speeded up by using a microprogrammable processor and microprogramming the simulation algorithm. For an Am29116 16-bit microprogrammable microprocessor, a speed-up factor of 5 to 10 can be obtained over the Intel 8086. Assuming an Am29116 and taking a speed-up value of 7, the average time required to process a simple element will be $a = 26$ μs. The value of a for functional elements will be 30 to 50 times larger.

Until now it has been assumed that all the processors have the same number of active elements. Assume that the unbalance due to nonideal partitioning introduces 10 percent more active elements in a processor ($k = 1.1$) and a is 40 times larger for functional evaluations than for simple evaluations [$a_{(se)} = 26$ μs for simple evaluations and $a_{(fe)} = 1040$ μs for functional evaluations]. The processing time for simple evaluations becomes $t_{p(se)} = 28.6(N/n)$ μs. The processing time for functional evaluations becomes $t_{p(fe)} = 1144(N/n)$ μs.

The processing time per active element (t_p/N) as a function of the number of processors (n) in the multiprocessor simulator is plotted for simple and functional evaluations in Fig. 15. If there is only one processor, then the processing time per active element for one simu-

* Trademark of Intel Corporation.

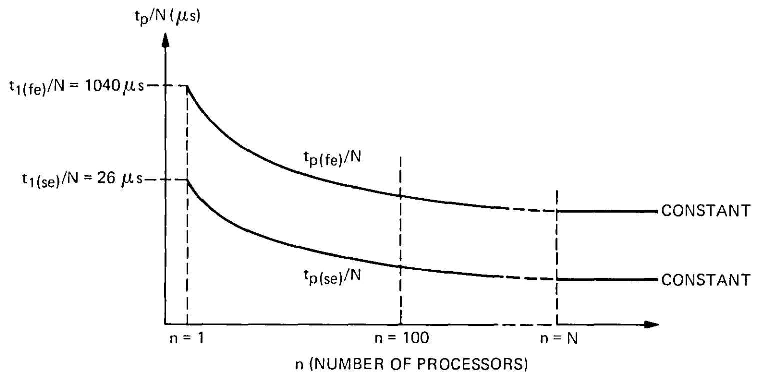


Fig. 15—Processing time as a function of number of processors for functional and simple evaluations.

lation cycle is $t_{1(se)}/N = 26$ μs for simple evaluations and $t_{1(fe)}/N = 1040$ μs for functional evaluations. Note that the unbalance factor k does not apply when there is only one processor.

The above analysis is done for $n \ll N$. For $n < N$ and $n \geq N$ there will be one active element per processor in the best case and the processing time per active element will remain constant at $t_p = (a/N)$ for all values of n greater than N (see Fig. 15). Also if $n \geq N$ and there is unbalance, then the value of k will be much larger.

6.4 Communication time t_c

The value of t_c will depend on the type of communication structure. Two types of communication structures will be considered here, a time-shared parallel bus and a cross-point matrix.

To estimate t_c for each case, first consider an element string yielded by the partitioning algorithm discussed previously (see Fig. 16). If f is the average fanout, one fanout line remains in the processor and $f - 1$ fanout lines go out to elements in other processors except the end of the string, where all fanout lines go to elements in other processors. Let c be the average number of elements in one string. Normally, one element per string is active during a simulation cycle. Define two adjacent element strings as two strings with common interconnections. Assume that all adjacent strings are in separate processors. This will give the situation that requires most communication and therefore the fastest communication structure. The average number of communication events generated by one active element during a simulation cycle that have to be sent over the communication structure is:

$$e = [f + (c - 1)(f - 1)]/c.$$

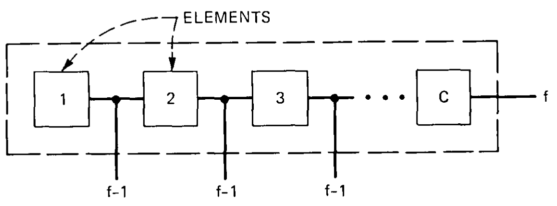


Fig. 16—An element string.

The typical value of f can be taken as 2 and for large circuits c is expected to be greater than 10. For f = 2 and c = 10, e will be equal to 1.1.

6.4.1 Time-shared parallel bus

Since the parallel bus is time-shared, the total communication time will be given by the time required to transmit one event multiplied by the number of communication events in a simulation cycle, i.e., Ne. The time required to transmit one event will be the sum of bus request and grant time (t_{brg}), address and data setup time (t_{ds}), data acknowledge time (t_{da}), and bus release time (t_{br}). An expression for the total communication time is:

$$t_{c(bus)} = (t_{brg} + t_{ds} + t_{da} + t_{br})(N)(e).$$

The address and data-setup time (t_{ds}) is composed of propagation delay without capacitance (t_{pd}) and capacitance delay (t_{cd}). These two parameters are functions of the bus length, which in turn will depend on the number of processors. If d is the distance between two processors then the average distance an event has to be sent is (nd)/2. Typical signal delay without capacitance is 1 ns/foot and if d is assumed to be 0.5 foot, then $t_{pd} = 0.25n$ ns. Capacitance will cause an extra delay of about 3 ns/foot giving $t_{cd} = 0.75n$ ns. The expression for $t_{c(bus)}$ becomes:

$$t_{c(bus)} = [n + (t_{brg} + t_{da} + t_{br})](N)(e) \text{ ns}.$$

Taking some typical values $t_{brg} = 100$ ns, $t_{da} = 50$ ns, $t_{br} = 50$ ns, and e = 1.1. The communication time per active element becomes:

$$t_{c(bus)}/N = (1.1n + 220) \text{ ns}.$$

This expression as a function of the number of processors in the multiprocessor simulator is plotted in Fig. 17a together with the expression for evaluation time per active element for simple evaluations and functional evaluations. Let t_1 be the length of the simulation cycle for a single-processor simulator and t_m be the length of the

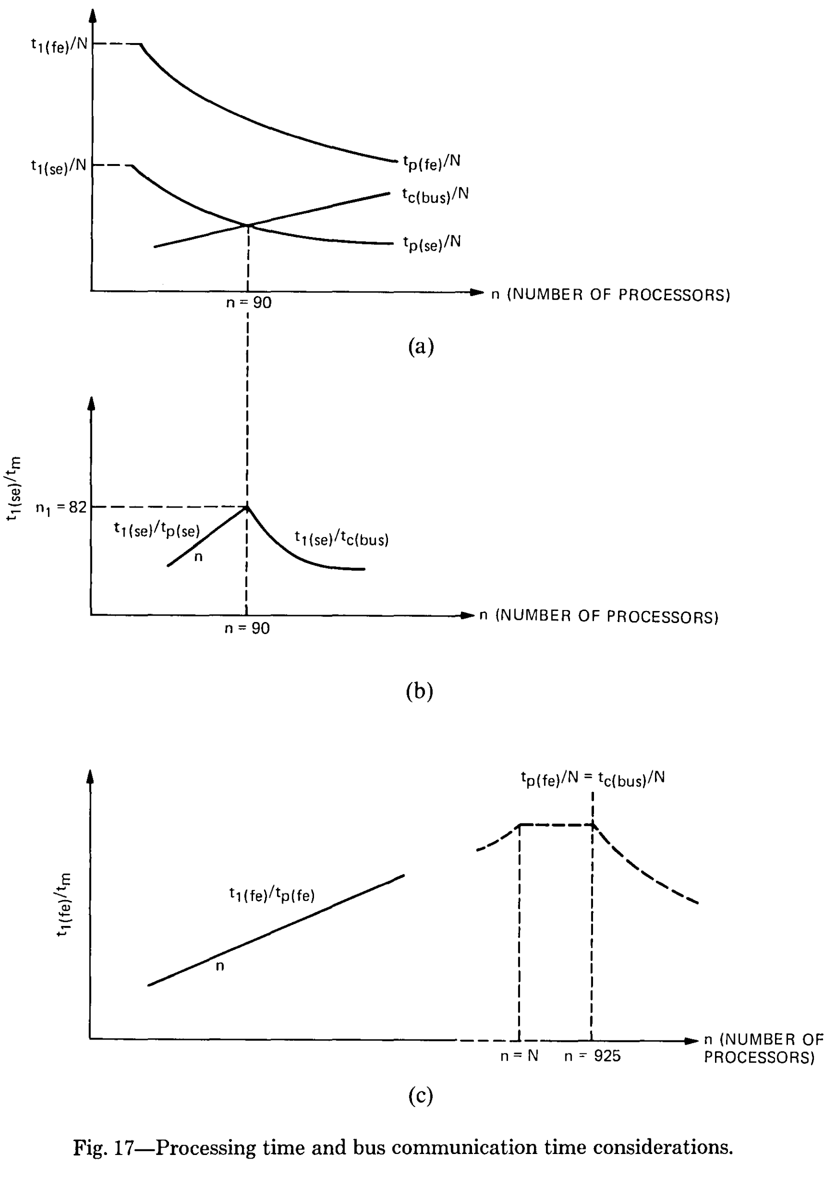


Fig. 17—Processing time and bus communication time considerations.

simulation cycle for a multiprocessor simulator. The performance of the multiprocessor simulator is defined as the ratio of single-processor time to multiprocessor time: t_l/t_m. The ratio of t_l to t_m is plotted in Fig. 17b for simple evaluations and in Fig. 17c for functional evaluations.

From Fig. 17b it can be seen that for simple evaluations the ratio t_l

to t_m is maximum for $t_{p(se)} = t_{c(bus)}$. For this condition, the multiprocessor simulator gives best speed performance over the single-processor simulator. The value for n is about 90 for this condition. Adding more processors will not speed up the simulator because the communication time will be a bottleneck.

Note also from Fig. 17b that for $t_c < t_p$ the processing time is a bottleneck and processors can be added (as long as $n \ll N$) to speed up the simulation time. For $t_c > t_p$, the communication time is a bottleneck, and a faster communication structure has to be added to speed up the simulation time. For best case $t_c = t_p$ and for further speed improvements, both faster processors (or more processors if $n \ll N$) and a faster communication structure must be added.

It is worthwhile noticing that the communication activity will be slightly lower than the predicted activity for smaller values of n than for larger values, since the probability that two adjacent element strings will be in the same processor is higher for smaller values of n. An element string is as defined in the section on circuit partitioning (Section 6.1), and two adjacent element strings are two strings with common interconnections. This is not expected to have any significant impact on the above analysis.

Figure 17c shows that for functional evaluations the processing time will be a bottleneck and the time-shared parallel bus provides the required communication speed as long as $n \ll N$.

A problem that may be encountered in the parallel bus structure is data skewing. The greater the number of lines on the communication bus, the greater the effect of data skew. Line conditioning in the form of bus extenders might be required for proper operation. Also, for large n, a hierarchical bus structure will be required for suitable operation.

6.4.2 Cross-point matrix

In a cross-point based communication structure, several processors can simultaneously send data to other processors. The total communication time will be governed by the processor having the maximum data to be sent over the communication structure, i.e. $(N/n)(k)(e)$ communication events. The time required to transmit one event will be the sum of the channel request and grant time (t_{crg}), delay incurred in transmission of message through matrix (t_{dm}), and channel release time (t_{cr}). Also when the processor asks to use the matrix, the destination processor might be busy. This requires selecting another event for transmission and trying to resend the blocked event at a later time. Let j be the number of events for which the channel is found busy and t_{rb} be the time wasted in processing a blocked request. An expression for the total communication time is:

$$t_{c(matrix)} = [t_{crg} + t_{dm} + t_{cr}](N/n)(k)(e) + (t_{rb})(j).$$

Taking some typical values $t_{crg} = 50$ ns, $t_{dm} = 100$ ns, $t_{cr} = 50$ ns, $k = 1.1$, $t_{rb} = 50$ ns, $e = 1.1$ and $j = 0.1(N/n)$ (the channel is found busy for 10 percent of the transfer requests). The communication time per active element becomes:

$$t_{c(matrix)}/N = 247/n \text{ ns}.$$

Comparing this value with $t_{p(se)}/N = 28.6/n$ μs and $t_{p(fe)}/n = 1144/n$ μs, it can be seen that the matrix will never cause a bottleneck in the multiprocessor simulator.

It is interesting to note that since the matrix provides a high-speed communication structure, some inefficiency in the partitioning algorithm can be tolerated. For $c = 1$ (smallest possible average chain) and a worst-case average fanout of 5, e will be equal to 5 and $t_{c(matrix)}/N = 1.105/n$ μs. This value is still much less than the processing time for simple evaluations.

Figure 18 shows the various processing and communication times. Once again, the above analysis has been done for $n \ll N$. For $n < N$ and $n \geq N$ the matrix communication time will remain constant.

6.5 Comparisons

Estimated values for t_c and t_p will now be derived. For realistic situations $1 \ll n \ll N$. For a circuit with 100,000 elements, assuming 2-percent activity per time frame, N will be 2,000.

For simple evaluations and a parallel bus, the maximum value of n is 90. For n greater than 90, the performance goes down since the bus becomes a bottleneck. For $n = 90$ the length of the simulation cycle is $t_{p(se)}/N = 28.6/n = 0.32$ μs. The number of evaluations per second becomes 3,125,000. This represents an increase by a factor of 100 over a traditional logic simulator (30,000 evaluations per second). The growth of the multiprocessor with parallel bus is restricted in the sense that this is the optimum performance and increasing the processors will not yield any further improvements. For modularity and greater performance, a matrix based communication structure is required for simple evaluations. With a cross-point matrix, the performance can be increased by increasing n as long as n remains much less than the activity N. With $n = 256$ and a cross-point matrix, the number of simple evaluations per second becomes 9,000,000. This represents an increase by a factor of 300 over the traditional logic simulator. For $n = 512$ and a cross-point matrix, the number of simple evaluations per second becomes 18,000,000. This represents an increase by a factor of 600 over a traditional logic simulator.

The speedup for functional evaluations with a time-shared parallel bus is of the same order. The maximum value of n in this case is 925. For $n = 90$, the speedup factor is 100 and for $n = 256$ the speedup

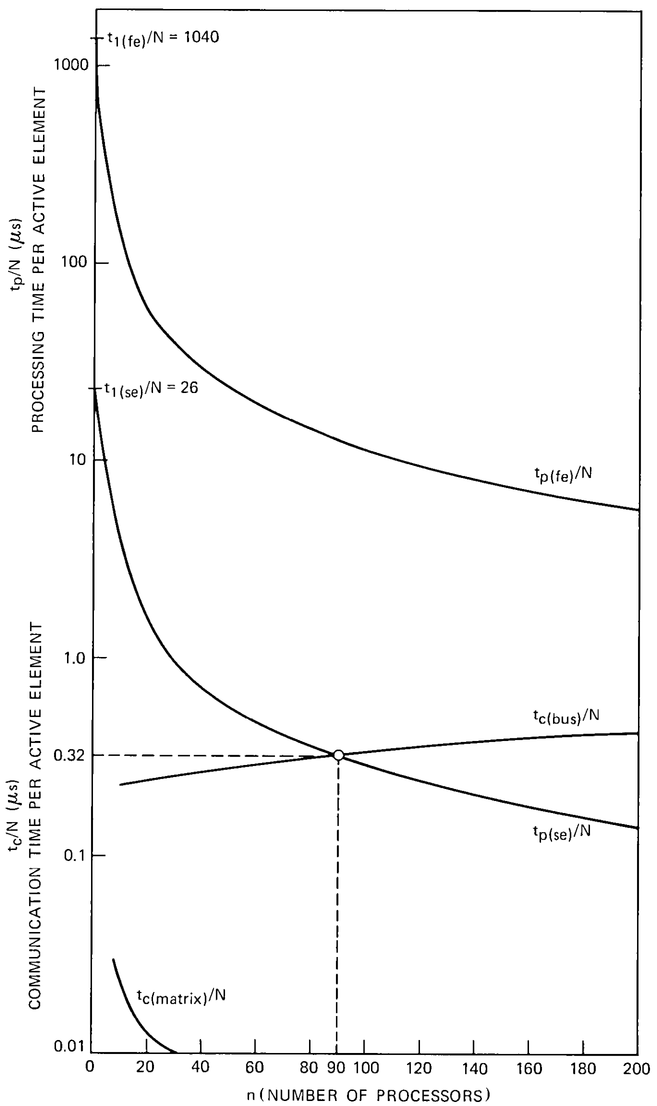


Fig. 18—Comparison of various communication times and processing times.

factor is 300. However, smaller values of n will be used in practice since the activity will be much lower for functional evaluations.

If slightly lower performance can be tolerated, then a nonmicroprogrammable microprocessor can be used. For the Intel 8086, the element-evaluation time becomes a = 181.5 μs. For a parallel bus, the value of n for optimum performance is 320. The number of simple evaluations per second becomes 1,600,000, an increase in performance of 53 over a traditional logic simulator. Once again, however, this is the best performance that can be obtained using the parallel bus. Using a cross-point matrix with n = 512, the speed up over the traditional logic simulator is 90.

VII. ARCHITECTURE CHOICE

It is seen from the previous section that a cross-point matrix is preferable for simulating a circuit containing simple elements. A time-shared parallel bus can be used for simple evaluations, but a speed penalty will be incurred and modularity will be lost. A time-shared parallel bus is sufficient for functional evaluations. A combination of the cross-point matrix and parallel bus can be used to simulate circuits containing both simple and functional elements.

For both simple evaluations and functional evaluations, a communication structure consisting of both a cross-point matrix and a time-shared parallel bus will prove cost effective. For transferring data between the cross-point matrix and the parallel bus a Bus Interface Unit (BIU) is required. The configuration of the BIU is shown in Fig. 19.

Data sequencer 1 transfers data from functional evaluators connected to the parallel bus to the simple evaluators and the master connected to the cross-point matrix. The data sequencer receives signal information and data from the parallel bus and transforms it for

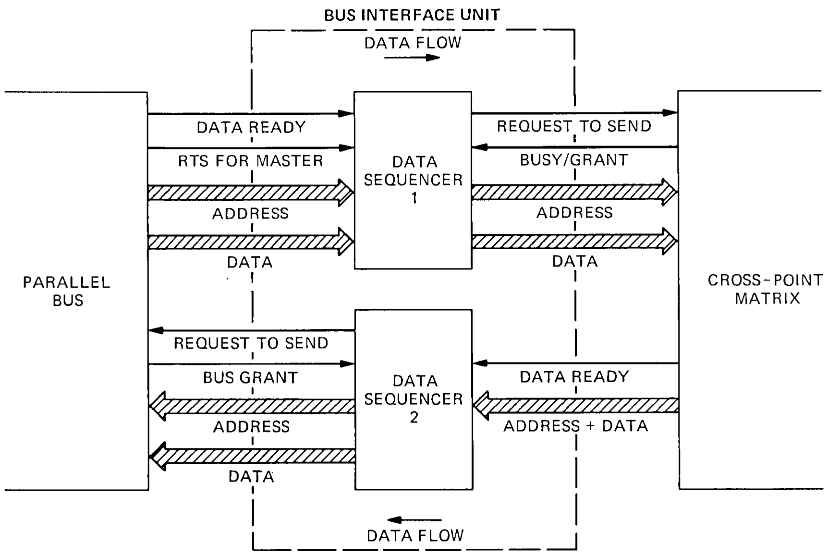


Fig. 19—Interface between parallel bus and cross-point matrix.

suitable transmission over the cross-point matrix. The parallel data received from the bus is transferred serially over the matrix.

Data sequencer 2 sends data from the cross-point matrix to the parallel bus. The serial data received from the cross-point matrix is transferred to parallel data for suitable transmission over the parallel bus.

VIII. PERFORMANCE ANALYSIS

The effect of varying various parameters on the performance of the multiprocessor is considered in this section. The analysis so far has been done using average values. The performance of the multiprocessor will be affected to some extent by variations in various parameters. The processing times $t_{p(se)}$ and $t_{p(fe)}$ are functions of N, n, k and a. The communication time for bus $t_{c(bus)}$ is a function of N, n, f, c. The communication time for matrix $t_{c(matrix)}$ is a function of N, n, k, f, c. The effect of variations in some of these parameters on performance of the logic simulator will be investigated.

8.1 Effect of changes in circuit activity N

8.1.1 Bus architecture

8.1.1.1 Simple evaluations. As seen from Fig. 20a, if activity N increases then $t_{p(se)}$ and $t_{c(bus)}$ increase proportionately. The operating point shifts from (1) to (2) for increasing values of N. This means that the length of the simulation cycle for the proposed simulator will increase by the same percentage that N increases. In the case of a single processor simulator, the simulation cycle will also increase by the same percentage. The performance index of the multiprocessor simulator (in terms of single processor time to multiprocessor time ratio) is not affected by the circuit activity, N (Fig. 20b).

8.1.1.2 Functional evaluations. The processing time of the multiprocessor simulator [$t_{p(fe)}$] is proportional to the processing time for simple evaluations [$t_{p(se)}$] by a slow-down factor of between 30 and 50. The analysis for simple evaluations done above, therefore, also applies for functional evaluations (i.e., the performance index of the multiprocessor simulator is not affected by circuit activity).

8.1.2 Matrix architecture (simple evaluations)

Since the processing time [$t_{p(se)}$] and matrix communication time [$t_{c(matrix)}$] are both proportional to N, they will increase by equal proportions. The length of the simulation cycle is governed by $t_{p(se)}$ and the performance index (single processor to multiprocessor time) is given by $t_l/t_m = (Na)/[(N/n)ka] = n/k$. Once again this value is independent of N and the performance index of a multiprocessor simulator with matrix architecture is not affected by circuit activity.

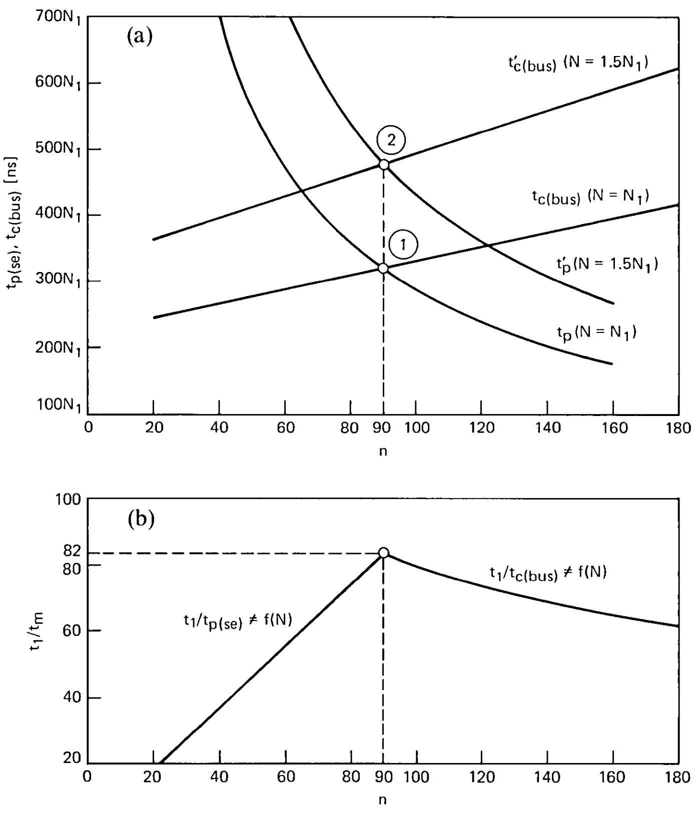


Fig. 20—Effect of changes in circuit activity. (a) Processing time (simple evaluations) and bus communication time for different values of N. (b) Performance variation for (a).

8.2 Effect of partitioning: variations in k

8.2.1 Bus architecture

8.2.1.1 Simple evaluations. The factor k represents the increase in activity in a processor, over the average N/n, due to non-ideal partitioning. An increase or decrease in k changes $t_{p(se)}$ in proportion but does not affect $t_{c(bus)}$. As seen from Fig. 21a for k = 1.1, the operating point is at (1) with n = 90. If k increases then the operating point moves to (2). The length of the simulation cycle increases in proportion to the increase in k. Since the length of the simulation cycle for the single-processor simulator is not affected by k, the multiprocessor performance index goes down. For n = 90, the performance index of the multiprocessor goes down from 82 to 64 as k increases from 1.1 to 1.4.

If the multiprocessor simulator is designed for a higher value of k, then the number of processors required for maximizing the perform-

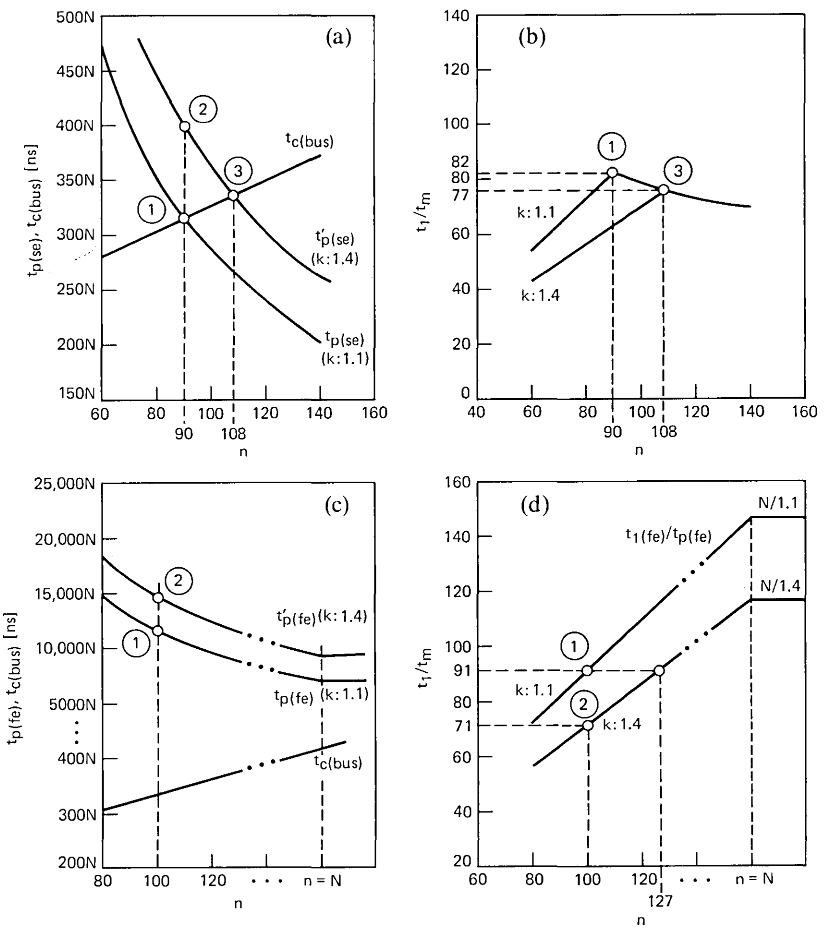


Fig. 21—Effect of partitioning: variations in k. (a) Processing time (simple evaluations and bus-communication time for various values of k. (b) Performance variation as affected by changes in k for (a). (c) Processing time (functional evaluations) and bus-communication time for various values of k. (d) Performance variation as affected by changes in k for (c).

ance index is greater than 90 and the maximum performance index decreases as k increases. This is shown by points (1) and (3) in Fig. 21b. For a multiprocessor simulator designed for k = 1.1, the maximum performance index is 82 for n = 90. If the multiprocessor is designed for k = 1.4, the maximum performance index is 77.

8.2.1.2 Functional evaluations. For functional evaluations, the curves for $t_{p(fe)}$ and $t_{c(bus)}$ meet for a large value of n (= 925). The length of the simulation cycle will be governed by the processing time. Figure 21c shows that for n = 100, the operating point will move from (1) towards

(2) as k increases from 1.1 to 1.4. Thus, the length of the simulation cycle will increase. The increase will not affect the single-processor simulator since its simulation cycle is not affected by k. Thus, the overall performance index will decrease as the activity increases. The performance index decreases from 91 to 71 as k increases from 1.1 to 1.4 for n = 100 (see Fig. 21d).

If the multiprocessor is designed for a higher value of k, then the number of processors required to maintain the performance index constant goes up. For example, for k = 1.1 and n = 100 the performance index is 91. If the simulator is to be designed for k = 1.4, then 127 processors are required to maintain the performance index at 91.

8.2.2 Matrix architecture (simple evaluations)

Once again, the processing time [$t_{p(se)}$] and matrix communication time [$t_{c(matrix)}$] are proportional to k. The processing time will determine the length of the simulation cycle regardless of the value of k. This case is similar to that in the previous section for functional evaluations and a parallel bus (Section 8.2.1). The overall performance index will go down as the activity increases. Also if the multiprocessor is designed for a higher value of k then the number of processors required to maintain the performance index constant goes up.

8.3 Effect of partitioning: variations in c

8.3.1 Bus architecture

8.3.1.1 Simple evaluations. Variations in c, the average number of elements per element string, will affect the bus communication time $t_{c(bus)}$. A decrease in c will increase the bus communication time. Figure 22a shows that the operating point moves from (1) to (2) as the value of c decreases from 10 to 1. For a constant number of processors and a decreasing c, the communication time may become a bottleneck. The length of the simulation cycle will increase and the performance index will go down.

If the simulator is designed for smaller values of c, the number of processors required will be smaller but the maximum performance goes down [operating point (3) in Fig. 22b].

8.3.1.2 Functional evaluations. For functional evaluations, the processing time is a bottleneck. The worst-case value of c is one (i.e., chains of length one). Even for this case the curves for $t_{p(fe)}$ and $t_{c(bus)}$ meet for a large value of n (= 663). Thus, the simulator for functional evaluations is not affected by c (typical value of n is 100).

8.3.2 Matrix architecture (simple evaluations)

As noted at end of Section 6.4.2, the processing time $t_{p(se)}$ will still be

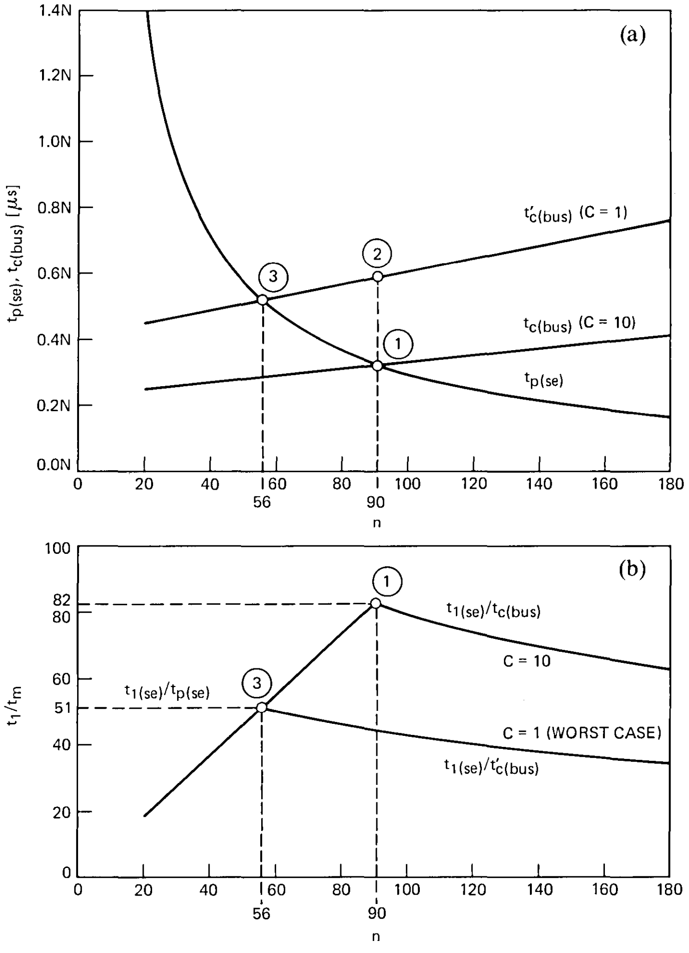


Fig. 22—Effect of partitioning: variations in c. (a) Processing time (simple evaluations) and bus-communication time for different values of c. (b) Performance variation as affected by changes in c for (a).

a bottleneck for worst-case value of c = 1. Thus, the performance of the simulator with a matrix is not affected by variations in c.

8.4 Effect of variations in fanout, f

8.4.1 Bus architecture

8.4.1.1 Simple evaluations. An increase in f will increase the bus communication time $t_{c(bus)}$. The processing time will not be affected by changes in f. Figure 23a shows that the operating point moves from (1) to (2) as fanout increases. Running circuits with larger fanout on a

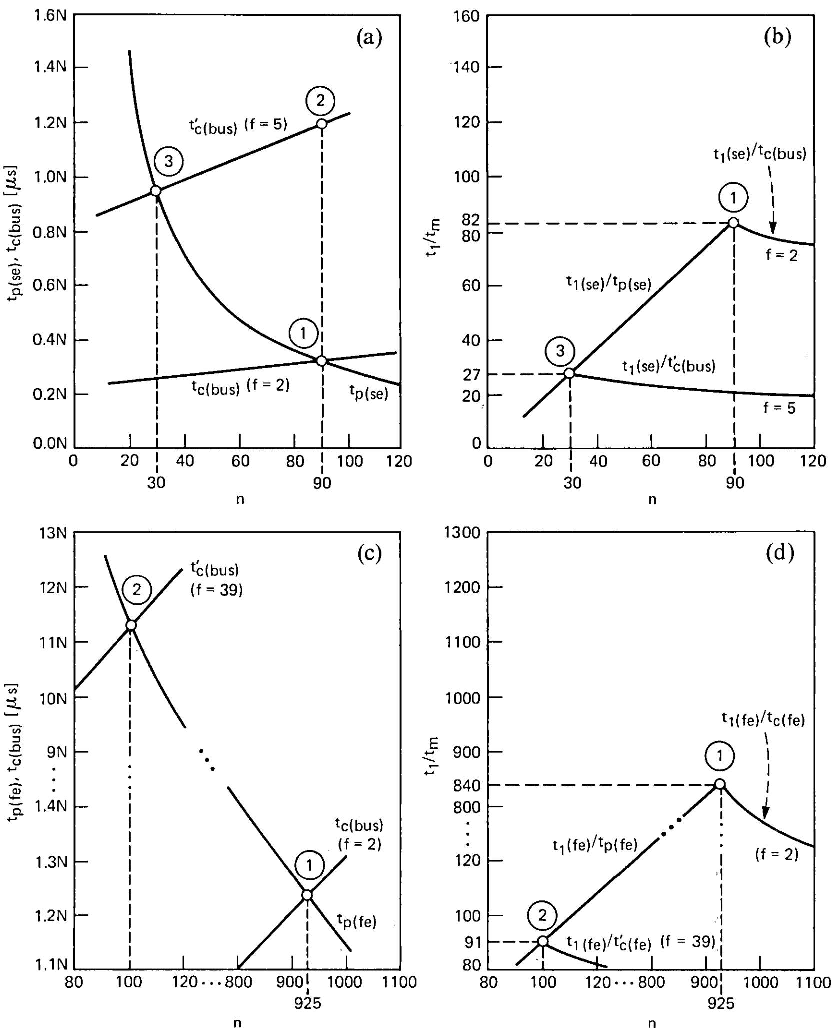


Fig. 23—Effect of variations in fanout. (a) Processing time (simple evaluations) and bus-communication time for different values of f. (b) Performance variation as affected by changes in f for (a). (c) Processing time (functional evaluations) and bus-communication time for different values of f. (d) Performance variation as affected by changes in f for (c).

simulator designed to operate with an average fanout of 2 will cause the communication time to become a bottleneck. Since the processing time does not change, the performance index of the simulator will decrease. Similarly, if a simulator is designed for larger fanout its maximum performance index would be less than that of a simulator designed for a smaller number of fanout. (See Fig. 23b.)

8.4.1.2 Functional evaluations. The simulator for functional evaluations is sensitive to changes in f. As seen from Fig. 23c, for $n = 100$ and $f > 39$ the communication time becomes a bottleneck and the performance index of the simulator goes down. The maximum performance index of the simulator is also lower for higher fanout. For large fanout, the communication time becomes a bottleneck for functional evaluations and a matrix may have to be used.

8.4.2 Matrix architecture

8.4.2.1 Simple evaluations. For an average fanout of 2, the processing time is a bottleneck. As the fanout increases, the curves for $t_{p(se)}$ and $t_{c(matrix)}$ approach each other. The communication time becomes a bottleneck for $f > 128$. This is an unrealistically large value and will not occur in practice.

8.4.2.2 Functional evaluations. The effect on functional evaluations is the same as discussed above, but $f > 5{,}100$.

IX. SUMMARY

The architecture of a multiprocessor simulator has been presented. The speed/performance ratio of the simulator is expected to be greater than two orders of magnitude compared to traditional simulation methods implemented on general-purpose computers. The power of the simulator can be increased over a certain range by increasing the number of slaves. Also the cost of the CPU time should be much lower than that obtained from general-purpose computers.

The architecture presented in this paper is expected to be faster than those of Barto and Szygenda, and Abramovici et al. The Yorktown Simulation Engine (YSE) built by IBM is reported to be faster than the architecture presented here, but the cost of the machine would be substantially higher since it uses special-purpose hardware. The architecture presented here and the YSE both try to take advantage of logic circuit concurrency to improve simulation performance. Unlike the YSE, our architecture implements event-driven simulation and is applicable to simulation with arbitrary delays at both gate and functional levels. Further work to be done includes detailed comparison of the various architectures in terms of performance and cost.

The application of the multiprocessor simulator to fault simulation is being investigated at the present time.

X. ACKNOWLEDGMENTS

The authors wish to thank J. Grason and D. B. Armstrong for reviewing this paper and providing helpful suggestions.

APPENDIX A

Partitioning Algorithm

A way to partition the logic circuit is to first generate and assign element strings to blocks, where an element string is as defined in Section 6.1. The next step is to balance the load (number of elements) in the blocks such that all blocks have approximately the same number of elements. Note that the elements have to be ordered according to levels prior to partitioning of the logic circuit. The partitioning algorithm is detailed below:

A.1 Generate and assign strings

1. $i = 0$
   [Note: a_i is the block to which assignments are currently being made.]
2. Select an unmarked element whose fanins have all been assigned previously to blocks other than a_i. Call the selected element e_s. If there is no such element and there are still some unmarked elements, go to step 6. If all elements are marked, go to algorithm A.2 (Load Balancing).
   [Note: Primary inputs can be treated as elements whose fanin has been previously assigned to blocks other than a_i.]
3. Assign the selected element e_s to block a_i and mark e_s.
4. If any fanout e_k of e_s is in a_i, go to 6.
   [Note: If a fanout element has been previously assigned to the current block a_i, then assigning another fanout element to the current block will require sequential processing of two elements.]
5. If there is an unmarked fanout e_k of e_s such that no fanin (e_k) (except e_s) is in a_i,
   then: (a) $s = k$
   (b) Go to step 3.
   [Note: If all of the fanout elements have been assigned to some other blocks, then the string cannot be extended. Note that primary outputs can be treated as being assigned to a null block.]
6. (i) $i = (i + 1)(\text{modulo } n)$
   (ii) Go to step 2.

A.2 Balance load

1. Total the number of elements in each block.
2. a_{max} = block with most elements
   a_{min} = block with fewest elements

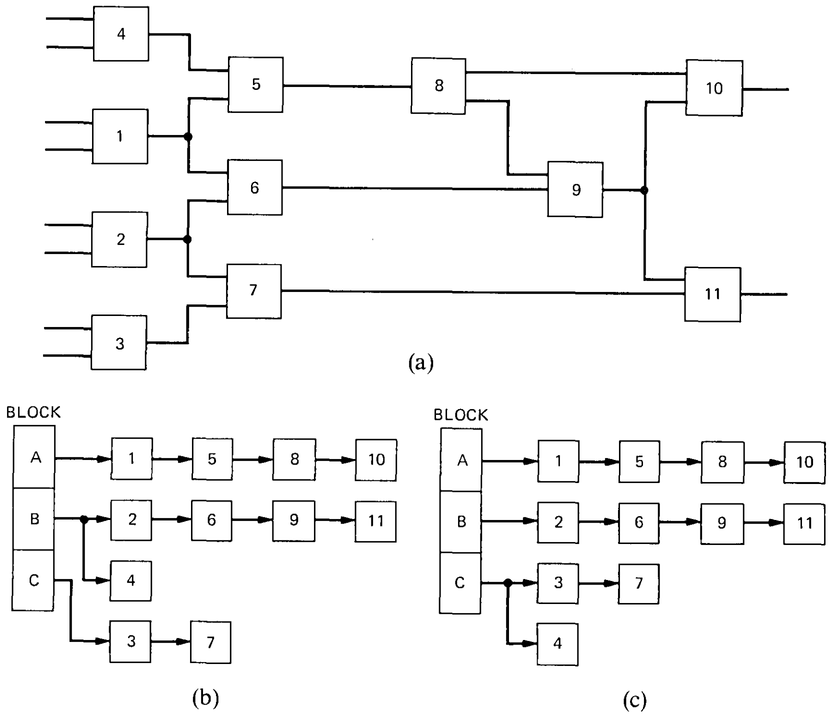


Fig. 24—Partitioning example. (a) Logic circuit to be partitioned. (b) String assignments before load balancing. (c) String assignments after load balancing.

e_{max} = number of elements in a_{max}
e_{min} = number of elements in a_{min}

3. If $(e_{max} - e_{min}) < [(\text{total number of elements})/n]0.1$, stop. [Note: If the maximum unbalance is less than 10%, stop.]
4. Select string s_i in a_{max} such that length $(s_i) < e_{max} - e_{min}$.
5. Move string s_i to block a_{min}.
6. If $(e_{max} - e_{min}) < [(\text{total number of elements})/n]0.1$, go to step 4. Else: Go to step 2.

As an example, consider the circuit in Fig. 24 to be partitioned into three blocks. The blocks and strings are then assigned as shown while looping through steps 2 through 6 in Section A.1. Note that element 4 is not assigned to block A since the fanout element 5 is already in block A (step 4 in Section A.1.). At end of load balancing element 4 will be assigned to block C.

Note that zero delay elements are not considered in the algorithm. Any zero delay element and all of its fanouts must be in the same block of the partition.

APPENDIX B

Glossary

BIU—Bus Interface Unit.

CPU—Central Processing Unit (part of the master unit).

EOD—End of Data (flag).

IDS—Input Data Sequencer.

IFB—Input FIFO Buffer.

L_t—Current list of timing wheel.

N—Average number of active elements per simulation cycle.

ODS—Output Data Sequencer.

OFB—Output FIFO Buffer.

RTS—Request to Send line.

RTSM—Request to Send to Master.

PU—Processing Unit. (Part of a slave unit)

a—Time required to process one active element.

a_i—A block of a partitioned circuit.

b_{ij}—Signal connections between blocks a_i and a_j.

c—Average number of elements in an element string.

c_i—Subcircuit located in processor p_i.

d_{ij}—Data path between processors p_i and p_j.

f—Average fanout of an element.

k—Imbalance factor owing to non-ideal partitioning.

n—Number of processors in the multiprocessor simulator.

p_i—Processor in multiprocessor simulator.

t_1—Length of simulation cycle for a single processor simulator.

$t_{1(fe)}$—Processing time during one simulation cycle with single processor (functional evaluations).

$t_{1(se)}$—Processing time during one simulation cycle with single processor (simple evaluations).

t_{br}—Bus release time for a parallel bus structure.

t_{brg}—Bus request and grant time for a parallel bus structure.

t_c—Total communication time during one simulation cycle.

$t_{c(bus)}$—Total communication time during one simulation cycle for a parallel bus structure.

$t_{c(matrix)}$—Total communication time during one simulation cycle for a matrix structure.

t_{cd}—Capacitance delay portion of address and data setup time, t_{ds}.

t_{cr}—Channel release time for a matrix structure.

t_{crg}—Channel request and grant time for a matrix structure.

t_{da}—Data acknowledge time for a parallel bus structure.

t_{dm}—Delay incurred in transmission of message through matrix.

t_{ds}—Address and data setup time for a parallel bus structure.

t_m—Length of simulation cycle for a multiprocessor simulator.

t_p—Average processing time per processor during a simulation cycle, where average processing time consists of the time required to process all active elements and schedule resulting events.

$t_{p(fe)}$—Average processing time per processor during one simulation cycle for functional evaluations.

$t_{p(se)}$—Average processing time per processor during one simulation cycle for simple evaluations.

t_{pd}—Propagation delay portion (without capacitance) of t_{ds}.

REFERENCES

1. M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems,* Rockville, Maryland: Computer Science Press, 1976, p. 148.
2. E. W. Thompson and S. A. Szygenda, "Digital Logic Simulation in a Time-Based, Table-Driven Environment: Part 2. Parallel Fault Simulation," Computer, *8-3* (March 1975), pp. 38–49.
3. E. G. Ulrich and T. G. Baker, "Concurrent Simulation of Nearly Identical Networks," Computer, *7-4* (April 1974), pp. 39–44.
4. D. B. Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits," IEEE Trans. Comp., *C-21,* No. 5 (May 1972), pp. 464–71.
5. R. Barto and S. A. Szygenda, "A Computer Architecture for Digital Logic Simulation," Elec. Eng., *55,* No. 642 (September 1980), pp. 35–66.
6. M. Abramovici, Y. H. Levendel, and P. R. Menon, "A Logic Simulation Machine," Proc. 19th Design Automation Conf., Las Vegas, Nevada, June 14–16, 1982, pp. 65–73.
7. G. Pfister, "The Yorktown Simulation Engine: Introduction," Proc. 19th Design Automation Conf., Las Vegas, Nevada, June 14–16, 1982, pp. 51–54.
8. M. M. Denneau, "The Yorktown Simulation Engine," Proc. 19th Design Automation Conf., Las Vegas, Nevada, June 14–16, 1982, pp. 55–59.
9. E. Kronstadt and G. Pfister, "Software Support for the Yorktown Simulation Engine," Proc. 19th Design Automation Conf., Las Vegas, Nevada, June 14–16, 1982, pp. 60–64.
10. N. D. Phillips and J. G. Tellier, "Efficient Event Manipulation, A Key to Large Scale Simulation," Proc. IEEE 1978 Semiconductor Test Conf., Cherry Hill, New Jersey, October 31–November 2, 1978, pp. 266–73.
11. J. G. Vaucher and P. Duval, "A Comparison of Simulation Event List Algorithms," Comm. ACM, *18-4* (April 1975), pp. 223–30.
12. J. Feder, unpublished work.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

The Limit of the Blocking As Offered Load Decreases With Fixed Peakedness

By P. J. BURKE

(Manuscript received April 14, 1982)

For a simple overflow process (SOP) with intensity a *and fixed peakedness* z *offered to a blocking system with* N *trunks, it is shown that the blocking approaches* $(1 - 1/z)^N$ *as* a *approaches zero. Further, for* $N > 1$ *and* a *sufficiently small, it is shown that the blocking increases as* a *decreases, while for* $N = 1$ *the blocking decreases monotonically. This result is helpful in restricting the range of use of algorithms based on simple overflow processes; for* $N > 1$ *when blocking less than the limit is encountered, nonmonotone behavior of the blocking function can be expected as the offered load is reduced. The present result is closely related to one of A. A. Fredericks, who found that the peakedness of the overflow from* N *trunks offered an SOP with intensity* a *and fixed peakedness* z *approaches* z *as* a *approaches zero. To give an intuitive explanation of Fredericks' result and the present one, it is shown that both would follow if the SOP asymptotically behaves as a thin batch process in which the batch size has a geometric distribution.*

I. INTRODUCTION

A blocking system is a trunk group with no queueing; when an arriving call does not find an idle trunk immediately, it overflows from the system. Every trunk group mentioned here is assumed to be a blocking system and to have independent exponential service times with unit mean. Also, statistical equilibrium is always assumed. If the arrival stream to a trunk group is Poisson, the stream of overflow calls is termed a simple overflow process (SOP). An SOP is characterized by two parameters; generally, these are taken to be the intensity and peakedness, denoted here as a and z, respectively. (We follow Fredericks[1] for terminology and notation as well as for proofs or references not given here.)

A trunk group with SOP input is the model underlying the Equivalent Random method[2] of sizing trunk groups that carry traffic whose peakedness is greater than unity. Algorithms based on the Equivalent Random method, or on approximations to it, are used to relate trunk-group size to the parameters of the input traffic to achieve a given probability of blocking (blocking). When these algorithms iterate on a, with z (and possibly other parameters) fixed, to achieve a low blocking on a given trunk group, they have been found sometimes not to converge, most prominently for large values of z.[3] This will certainly happen if there exists a minimum blocking, over all values of a, for the given values of the other parameters, and the target blocking is less than this minimum. An algorithm may fail also if it requires that the blocking be a monotone function of a when relevant values of a are in a region where this function is nonmonotone. If the blocking has a positive limit as a approaches zero and also has a unique minimum, one can avoid algorithmic failure owing to these causes by avoiding blocking values less than the limit, or at least treating them with caution. In what follows we find the limit of the blocking as a decreases to zero, which is positive for z greater than unity; and, although we do not prove the uniqueness of the minimum, we show that the limit is approached from below for all trunk groups with more than one trunk.

II. THE LIMIT OF THE BLOCKING

When an SOP is offered to a group of N trunks, the blocking on this group is denoted as $B(N, a, z)$. When the offered stream is Poisson, the blocking is written $B(N, a)$, given by the well-known formula

$$\begin{aligned} B(N, a) &= e^{-a}a^N \Big/ \int_a^\infty e^{-x}x^N dx, \\ &= (a^N/N!) \Big/ \sum_{i=0}^{N} a^i/i! \end{aligned}$$

for N integral.

The following is true:

$$\text{For } z \text{ fixed,} \qquad \lim_{a \to 0} B(N, a, z) = (1 - 1/z)^N. \tag{1}$$

To prove (1) we note that there exist a Poisson load A and a (nonintegral) number of trunks T such that

$$\begin{aligned} a &= AB(T, A) \\ z &= A/(T + 1 + a - A) + 1 - a, \end{aligned} \tag{2}$$

in which case

$$B(N, a, z) = B(T + N, A)/B(T, A). \tag{3}$$

Because the case of N integral is particularly simple, we treat it separately. Since $B(T, A)$ is continuous (in both arguments), and as a decreases to zero, T increases monotonically without limit (for fixed $z > 1$), we can find a decreasing sequence of values of a for which T is integral; and thus for the present we treat T as an integer without further comment.

From (3) and using the formula for $B(T, A)$, after a little simplification,

$$B(N, a, z) = A^N \Big/ \left\{ (T+1)^{(N)} \left[1 + a \sum_{i=1}^{N} A^{i-1}/(T+1)^{(i)} \right] \right\}, \tag{4}$$

where $T^{(i)} = T(T+1) \cdots (T+i-1)$.

For the case $N = 1$, we show that $B(N, a, z)$ strictly decreases to its limit $1 - 1/z$ as $a \to 0$. From (4),

$$\begin{aligned} B(1, a, z) &= A/\{(T+1)[1 + a/(T+1)]\} \\ &= A/(T+1+a). \end{aligned} \tag{5}$$

From (2), we obtain

$$A/(T+1+a) = 1 - 1/(z+a). \tag{6}$$

From (6) it is clear that $B(1, a, z)$ strictly decreases to $1 - 1/z$ as $a \to 0$. Furthermore, we note from (6) that

$$A/(T+1) \to 1 - 1/z. \tag{7}$$

From (4), using (7),

$$B(N, a, z) \sim [A/(T+1)]^N \to (1 - 1/z)^N. \tag{8}$$

III. ASYMPTOTIC BEHAVIOR OF THE BLOCKING

In this section, the requirement that N be integral is relaxed, and we investigate the magnitude of $B(N, a, z)$ in the neighborhood of its limit.

For real N we write

$$\begin{aligned} B(N, a, z) &= \frac{A^N T!}{(T+N)!} \cdot \frac{\dfrac{1}{T!}\displaystyle\int_A^\infty e^{-x} x^T dx}{\dfrac{1}{(T+N)!}\displaystyle\int_A^\infty e^{-x} x^{T+N} dx} \\ &= \frac{A^N T!}{(T+N)!} \cdot \frac{Q[2A \,|\, 2(T+1)]}{Q[2A \,|\, 2(T+N+1)]}, \end{aligned} \tag{9}$$

where $Q = Q(\chi^2 \,|\, \nu)$ is the complementary χ^2 distribution function (d.f.) with ν degrees of freedom.

Since $A/(T+1) \rightarrow 1 - 1/z$ for $a \rightarrow 0$, both χ^2 d.f.'s will approach unity. An estimate of the rapidity of this approach will be needed below. First,

$$1 - Q[2A|2(T+1)] \sim P(x),$$

where

$$x = (A - T - 1)/\sqrt{T+1}$$

and $P(x)$ is the standard normal d.f.[4] Also, $P(x) = Q(-x)$, where $Q(x)$ is the complementary normal d.f., and it is known[5] that

$$\begin{aligned} Q(x) &\sim 1/(x\sqrt{2\pi})\exp(-x^2/2) \\ &= o\{1/(\sqrt{2\pi})\exp[-(T+1-A)^2/2(T+1)]\} = o(T^{-k}) \end{aligned}$$

for any finite k. Since we will be interested in expanding $\log B(N, a, z)$ only as far as infinitesimals of order T^{-1}, the χ^2 d.f.'s in (9) can be ignored. If we use Stirling's formula and (7),

$$\begin{aligned} B(N, a, z) &\sim A^N T^{T+1/2} e^{-T}/[(T+N)^{T+N+1/2} e^{-(T+N)}] \\ &\sim (A/T)^N/[(1+N/T)^{T+N+1/2} e^{-N}] \\ &\sim (1-1/z)^N (1+1/T)^N/[(1+N/T)^{T+N+1/2} e^{-N}]. \quad (10) \end{aligned}$$

Also,

$$\begin{aligned} \log B(N, a, z) &\sim \log(1-1/z)^N + N\log(1+1/T) \\ &\quad - (T+N+\tfrac{1}{2})\log(1+N/T) + N \\ &\sim \log(1-1/z)^N - (N^2-N)/2T. \quad (11) \end{aligned}$$

Therefore, for $N > 1$, $B(N, a, z)$ increases to its limit, $(1-1/z)^N$, for sufficiently small a. This indicates that the dip in the load-blocking curves for fixed z shown in Ref. 6 (pages 30 to 32) exists for every $N > 1$.

IV. AN ASYMPTOTIC STRUCTURAL CONJECTURE

To provide some intuitive understanding of his result that the peakedness of the overflow from N trunks approaches z as $a \rightarrow 0$, Fredericks says, "That is, by fixing z and letting $a \rightarrow 0$ we are charging the 'structure' of the process to one that is 'infinitely bunchy'." In an attempt to extend the provision of intuitive understanding of Fredericks' result and the present result, a conjectural structure is heuristically developed below for an SOP with a infinitesimal and z fixed.

Overflows from the equivalent group of T trunks, which constitute the SOP, occur only when that group is busy. The mean length of a busy period is $1/T$ and the mean number of overflowing calls during a busy period is $A/T \rightarrow 1 - 1/z$. With the mean time between overflows becoming infinite, the mean time between busy periods, or clusters of busy periods, must also become infinite. Thus, the SOP will consist of clusters of calls occurring during infinitesimal intervals (random numbers of closely spaced busy periods) separated by intervals that become infinitely large. The limiting structure appears to be that of a "thin" batch process.

Since we are concerned with the peakedness, we assume that the SOP is offered to an infinite trunk group and define the random variable X as the number of busy trunks at an arbitrary instant on such a group. We define a thin process as any call arrival process with an infinitesimal parameter λ, for which the generating function of the distribution of X may be written

$$G(s) = 1 - \lambda \int_0^\infty \{1 - \beta(e^{-t}s + 1 - e^{-t})\}dt + o(\lambda), \tag{12}$$

in which β is a probability generating function on the positive integers. (The generalization to other service-time d.f.'s is apparent.) The factorial-moment generating function corresponding to $G(s)$ is

$$F(s) = 1 - \lambda \int_0^\infty \{1 - \beta(e^{-t}s + 1)\}dt + o(\lambda), \tag{13}$$

and what might be called the factorial-cumulant generating function is

$$\log F(s) = -\lambda \int_0^\infty \{1 - \beta(e^{-t}s + 1)\}dt + o(\lambda). \tag{14}$$

Since the factorial cumulants bear the same relation to the factorial moments that the ordinary cumulants do to the ordinary moments, a comparison of (13) and (14) shows that for a thin process the cumulants are the same as the corresponding moments up to infinitesimals of higher order.

The mean of X is

$$m = \lambda b_1 + o(\lambda), \tag{15}$$

where b_1 is the mean batch size. The second factorial moment of X, from (13) or (14), is $\lambda b_2/2 + o(\lambda)$, where b_2 is the second factorial moment of the batch size. Hence, the peakedness of a thin SOP is

$$z = b_2/2b_1 + 1 + o(1). \tag{16}$$

If the probability of a batch of size x is pq^{x-1}, $q = 1 - p$, $x = 1$,

$\cdots$, then the probability that the batch size is $N + x$, given that it is at least $N + 1$, is

$$pq^{x+N-1} \Big/ \sum_{i=0}^{\infty} pq^{N+i} = pq^{x-1} \tag{17}$$

(the "lack of memory" of the geometric distribution). Since a thin input process results in a thin overflow process and since the peakedness of a thin process depends only on the batch-size distribution, Fredericks' result would follow if the SOP asymptotically is a thin geometric-batch process.

The first two factorial moments of a geometric distribution with no mass at zero are $b_1 = 1/p$, $b_2 = 2q/p^2$. Hence,

$$p = 1/z,$$

where, as in the following, infinitesimals are dropped. Next, we find the blocking met by a thin geometric-batch process offered to N trunks. Since asymptotically all the blocking derives from intra-batch interference, this is given by

$$B = p \sum_{i=1}^{\infty} q^{N+i-1} \cdot \frac{N+i}{1/p} \cdot \frac{i}{N+i}, \tag{18}$$

in which the last factor on the right-hand side is the conditional blocking probability, given that the call arrives in a batch of size $N + i$. The other factors together give the probability of such an arrival. (For a discussion see Ref. 7.)

Hence,

$$\begin{aligned} B &= p^2 q^N \sum_{i=1}^{\infty} iq^{i-1} \\ &= q^N = (1 - 1/z)^N, \end{aligned} \tag{19}$$

which is known to be correct.

REFERENCES

1. A. A. Fredericks, "Congestion in Blocking Systems—A Simple Approximation Technique," B.S.T.J., *59,* No. 6 (July–August 1980), pp. 805–27.
2. R. I. Wilkinson, "Theories for Toll Traffic Engineering in the U.S.A.," B.S.T.J., *35,* No. 2 (March 1956), pp. 421–514.
3. T. E. Ahern, unpublished results.
4. M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions,* U.S. Government Printing Office, 9th Printing (November 1970), Section 26.4.11.
5. *Ibid,* Section 26.2.12.
6. R. I. Wilkinson, *Nonrandom Traffic Curves and Tables,* Bell Laboratories, August 1970.
7. P. J. Burke, "Delays in Single-Server Queues with Batch Input," Oper. Res., *23,* No. 4 (July–August 1975), pp. 830–33.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

TV Bandwidth Compression Techniques Using Time-Companded Differentials and Their Applications to Satellite Transmissions

By K. Y. ENG and B. G. HASKELL

(Manuscript received December 21, 1981)

This paper describes various methods to compress the bandwidth of a National Television System Committee (NTSC) color TV signal. The methods are based on taking differences between adjacent or successive scan lines and then performing time companding on these differential signals. In the three specific methods outlined, a 4.2-MHz TV signal can be bandwidth-reduced to 3.34 MHz, 2.9 MHz, and 2.5 MHz, respectively, without loss of picture quality. When these techniques are used with time-compression multiplexing in satellite transmissions, high spectral efficiency can be obtained. Examples are shown where the transmission of two or three broadcast-quality TV's per 36-MHz transponder are possible with existing technology.

I. MOTIVATION

There has been considerable interest in recent years in transmitting two or more broadcast-quality TV signals through a single satellite transponder (usable bandwidth of 36 MHz). The difficulties of doing this with the conventional frequency-division-multiplexing (FDM) approach are due to the effect of transponder nonlinearities such as intermodulation and intelligible cross talk. To alleviate these problems, time-compression multiplexing (TCM) can be employed whereby the scan lines from different, but synchronized, TV signals can be time compressed and time multiplexed into the duration of an ordinary scan line for transmission via a single frequency-modulated (FM) carrier. The concept of TCM is not new,[1,2] but the recent commercial availability of fast analog/digital (A/D) converters, digital/analog (D/A) converters, charge-coupled devices (CCD), and memory devices makes it economically feasible for implementation. The bandwidth efficiency of TCM/FM outweighs that of FM/FDM whenever time division between different signals can be achieved more efficiently than frequency division. In the case of satellite transmission, this is generally true. Therefore, TCM is advantageous from both viewpoints of avoiding transponder nonlinearities and achieving higher spectral efficiency.

Haskell[3] recently demonstrated that a differential signal derived from two adjacent or successive scan lines of a National Television System Committee (NTSC) color TV has significantly lower bandwidth occupancy than the original 4.2-MHz composite signal. When this differential signal technique is used in combination with the concept of TCM, i.e., time companding (time compression or expansion), significant bandwidth compression on a TV can be obtained (see Sections II and III). When TCM is further exploited by application to different bandwidth-compressed TV signals, high spectral efficiency can be obtained in a satellite link. We discuss examples illustrating this idea in Section IV, where examples for two or three TV's per transponder are shown. Since all the techniques discussed are easy to implement with existing hardware, we conclude with an encouraging outlook into the foreseeable future when multiple high-quality TV transmissions through a single satellite transponder may become possible.

II. LINE AND FIELD DIFFERENTIAL SIGNALS

NTSC color TV pictures have an interlacing scan pattern whereby the scan lines are presented in the order of the odd-number lines (the odd field) first, followed by the even-number lines (the even field). The bandwidth of such a TV signal is assumed to be 4.2 MHz. A block diagram for generating a line differential signal is shown in Fig. 1. Consider an odd field where the scan lines are labeled 1, 3, 5, and so on. As line 1 arrives, it is stored. As line 3 arrives, it is stored while line 1 is read out simultaneously. When line 5 arrives, an output is formed

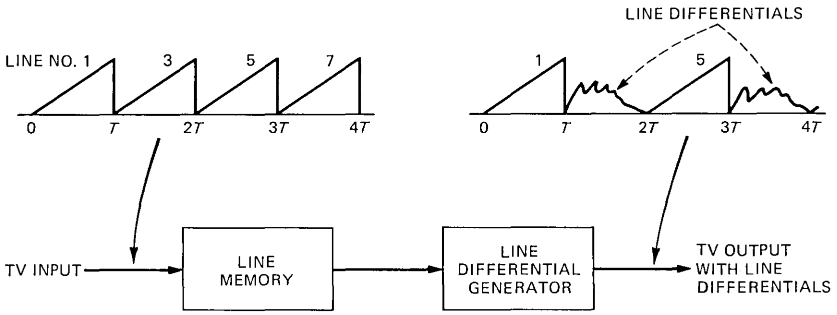


Fig. 1—Generation of a TV signal with line differentials.

consisting of line 3 minus some average of lines 1 and 5. This output is called a line differential signal. The operation is carried out for all subsequent lines. Therefore, the output of the line differential generator as shown in Fig. 1 consists of alternate lines of the original signal and a line differential signal. This composite output has a bandwidth of 4.2 MHz. However, if only the line differential signal is considered, its bandwidth can be reduced to 3 MHz without affecting picture quality.[3]

A block diagram illustrating the generation of a field differential signal is shown in Fig. 2. The scan pattern of the input TV signal is first changed from interlacing to sequential, i.e., line 1, line 2, line 3, ··· and so on. In doing so, there is at least a delay of one field for the signal. This sequential signal is processed as follows: As line 1 arrives, it is simultaneously stored and read out. When line 2 arrives, the difference between lines 1 and 2 is taken and is output in the duration following line 1. This difference signal is called the field differential signal because it is derived from two adjacent lines of two successive fields. The same operation is repeated for all successive pairs of input (sequential) lines. The resulting composite output consists of alternate lines of the original input signal and a field differential signal. This composite output again has a bandwidth of 4.2 MHz. If the field differential signal is considered by itself, the bandwidth can be reduced to 2 MHz without affecting picture quality.[3]

It should be pointed out that there are several ways to derive a line (or field) differential signal. One way is to take the difference between the current line and the average of the preceding and succeeding lines. This would, of course, involve more than a two-line memory, and hence, an additional delay. We digress from the detailed discussion on how to take these line and field differences.

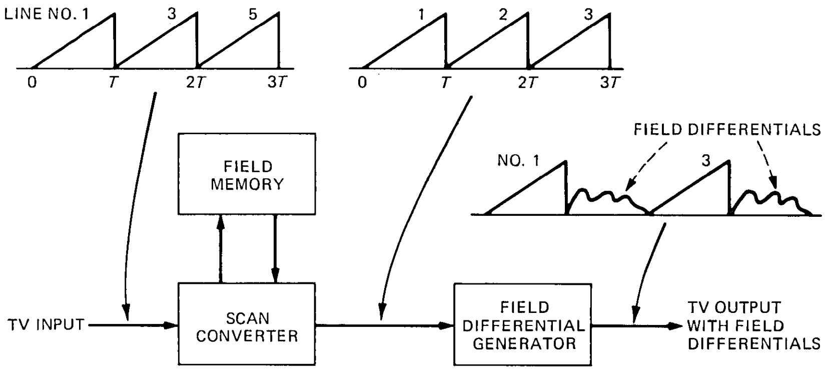


Fig. 2—Generation of a TV signal with field differentials.

III. BANDWIDTH COMPRESSION USING TIME COMPANDING

As we mentioned previously, it is straightforward to implement time companding (time compression or expansion) with readily available A/D and D/A devices (or CCD). The use of time companding in combination with the differential signals described in the preceding section can lead to significant bandwidth reduction in the TV signal. For instance, using time companding and the line differential signal the TV bandwidth can be reduced to 3.31 MHz. Similarly, it can be reduced to 2.9 MHz with the field differential signal, and to 2.5 MHz with the line plus field differential signals. Various ways to achieve these reductions are discussed below.

3.1 Line differential signal

For simplicity, let us restrict our attention to a pair of successive scan lines at the output of a line differential generator, say the intervals $(0, T)$ and $(T, 2T)$, where T is a scan line duration. As shown in Fig. 3, the interval $(0, T)$ contains the original 4.2-MHz signal; the interval $(T, 2T)$ contains the 3-MHz line differential signal whose horizontal blanking interval (about 17 percent of T) is identically zero and need not be transmitted. We perform a time expansion on the line $(0, T)$, resulting in an output in the interval $(0, T')$, $T' > T$ (neglecting actual delay for simplicity); and we perform a time compression on the input $(T, 2T)$, resulting in an output $(T', 2T)$. The time expansion factor α and compression factor β, both >1, are chosen such that the bandwidths of the signals in the output intervals $(0, T')$ and $(T', 2T)$ are equal. This can be written mathematically as

$$\frac{f_1}{\alpha} = \beta f_2, \tag{1}$$

where f_1 and f_2 are the maximum frequencies of the signals in the input

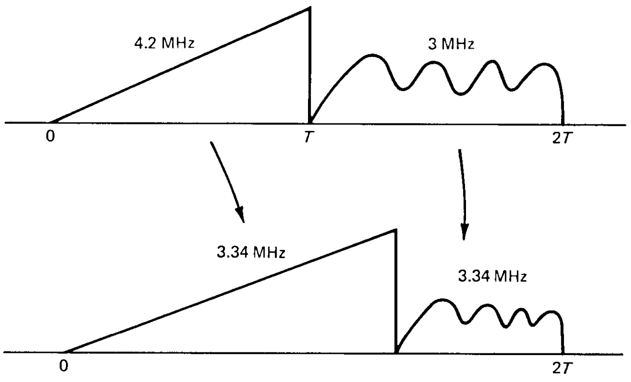


Fig. 3—Time companding of a line differential signal in $(0, T)$.

intervals (0, T) and (T, $2T$), respectively. Since the expanded and compressed line lengths are assumed to add up to $2T$, we have another constraint on α and β:

$$\alpha T + \frac{0.83T}{\beta} = 2T,$$

or

$$\alpha + \frac{0.83}{\beta} = 2. \tag{2}$$

Solving (1) and (2) with $f_1 = 4.2$ MHz and $f_2 = 3$ MHz, we obtain $\alpha = 1.255$ and $\beta = 1.115$. The bandwidth of the time companded output is given by (1), i.e.,

$$f_{\max} = \beta f_2 = 3.34 \text{ MHz}. \tag{3}$$

Therefore, the final output consists of successive pairs of scan lines. Within each of these pairs, one is a time-expanded version of the original line, and another is a time-compressed version of the line differential signal, resulting in a bandwidth reduction from 4.2 MHz to 3.34 MHz. Reconstruction of the original is accomplished by the inverse of the above operation.

3.2 Field differential signal

It is obvious that the above bandwidth reduction technique can also be used with the field differential signal in place of the line differential signal. The mathematical descriptions are the same, i.e., (1) and (2) also hold. With $f_1 = 4.2$ MHz and $f_2 = 2$ MHz, we obtain $\alpha = 1.433$ and $\beta = 1.465$. The maximum frequency of the final time-companded

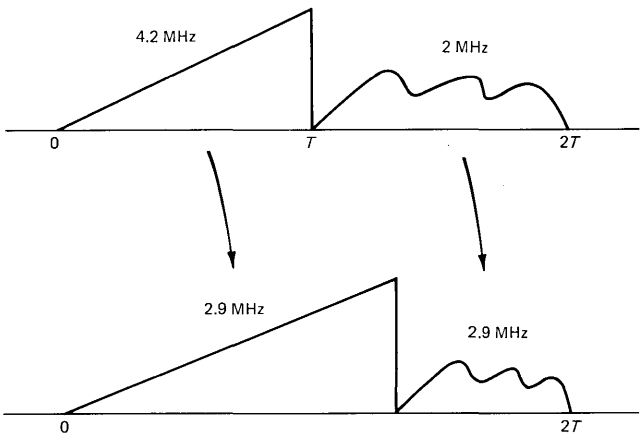


Fig. 4—Time companding of a field differential signal in (0, $2T$).

output is (see Fig. 4)

$$f_{\max} = \beta f_2 = 2.9 \text{ MHz}. \tag{4}$$

The use of the field differential signal thus reduces the TV bandwidth from 4.2 MHz to 2.9 MHz.

3.3 Line plus field differential signals

Further bandwidth reduction is yet possible with a combination of line plus field differential signals. We first convert the scan pattern of a 4.2-MHz input TV signal from interlacing to sequential. For easy understanding, we visualize that this sequential TV signal is to be processed by two tandem processors, say I and II (see Figure 5). Processor I works as follows: As line 1 arrives, it is simultaneously stored and read out. When line 2 arrives, a field differential signal derived from lines 1 and 2 is read out in the duration following line 1. This field differential signal can be bandlimited to 2 MHz without affecting picture quality, as previously mentioned, and its horizontal blanking interval (about 17 percent) is again not transmitted. As line 3 arrives, a line differential signal is taken for output. The line differential signal has a bandwidth of 3 MHz, and its horizontal blanking interval is not transmitted. When line 4 arrives, the field differential signal derived from lines 3 and 4 is read out. These operations in four consecutive line durations are repeated for all subsequent lines. As a result, processor I output consists of 4-line blocks, each of which contains a line of the original signal, a field differential signal, then a line differential signal, and finally another field differential signal. Within each of these 4-line blocks, processor II performs either time compression or expansion to each of the lines in a manner similar to that discussed above, so as to equalize the resulting bandwidths of the

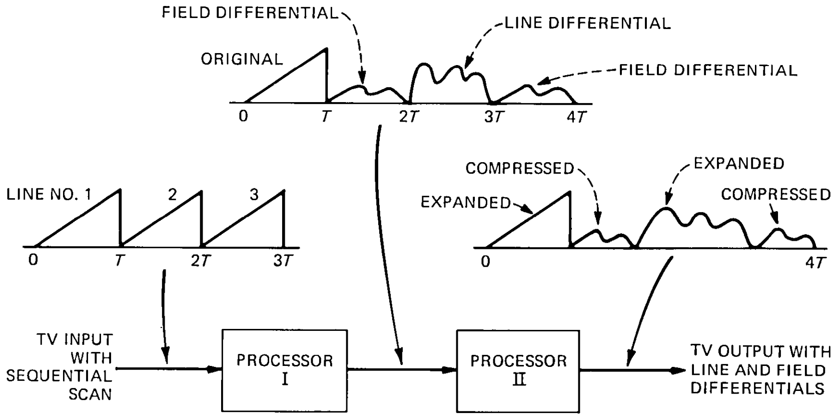


Fig. 5—Generation of a TV signal with line and field differentials.

four lines. It is briefly shown below that the bandwidth can be reduced from 4.2 MHz to 2.5 MHz.

Again, let T be a line duration (Fig. 6). In the interval (0, $4T$), we have one line of the original signal (f_1 = 4.2 MHz), a field differential signal (f_2 = 2 MHz), a line differential signal (f_3 = 3 MHz), and finally another field differential signal (f_4 = 2 MHz). The equations to equalize bandwidths and to adjust time durations are:

$$\frac{f_1}{\alpha} = \alpha_2 f_2 = \frac{f_3}{\alpha_3} = \alpha_4 f_4, \tag{5}$$

and

$$\alpha_1 T + \frac{0.83T}{\alpha_2} + 0.83\alpha_3 T + \frac{0.83T}{\alpha_4} = 4T, \tag{6}$$

where α_1 and α_3 are time-expansion factors for bandwidths f_1 and f_3, respectively; α_2 and α_4 are time-compression factors for bandwidths f_2 and f_4, respectively; and the factor 0.83 accounts for the deletion of the horizontal blanking interval. The solution for the above is: α_1 = 1.68, α_2 = 1.25, α_3 = 1.20, and α_4 = 1.25. The overall bandwidth is then

$$f_{\max} = \alpha_2 f_2 = (1.25)(2) \text{ MHz} = 2.5 \text{ MHz} \tag{7}$$

The preceding descriptions outline the system operations conceptually. In an actual implementation, the various functions of scan conversion, taking differences, and time companding need not be done separately as presented. For instance, once the input TV signal is digitized, all the functions can be realized in an integrated manner.

IV. APPLICATIONS TO MULTIPLE TV TRANSMISSIONS THROUGH SATELLITES

In this section we discuss some applications of the various bandwidth compression techniques (outlined in Sections II and III) to multiple TV transmissions through satellites. We assume that TCM is to be

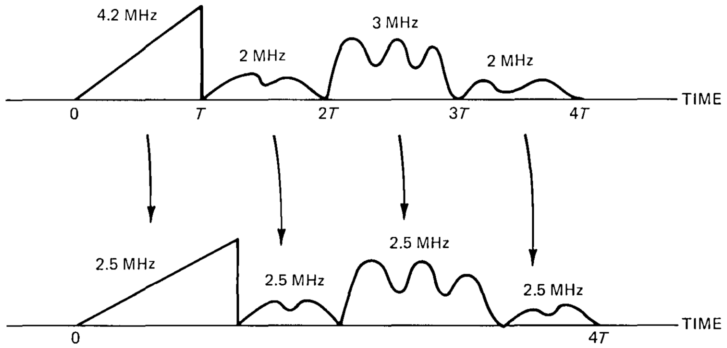


Fig. 6—Time companding of line and field differential signals in (0, $4T$).

employed for multiplexing different but synchronized TV signals before transmission via a single FM carrier. The performance of a simplified TCM/FM satellite link is shown in Fig. 7, where the following assumptions are made:

(*i*) Only the downlink is considered.

(*ii*) Time compression and expansion are ideal.

(*iii*) Satellite Effective Isotropic Radiated Power (EIRP) = 34 dBW; satellite usable channel bandwidth = 36 MHz; receive earth station elevation angle = 15°; frequency = 4180 MHz; maximum receive flux density = -152 dBW/m^2/4 kHz.

(*iv*) The peak-to-peak signal to weighted rms noise ratio (in dB) is

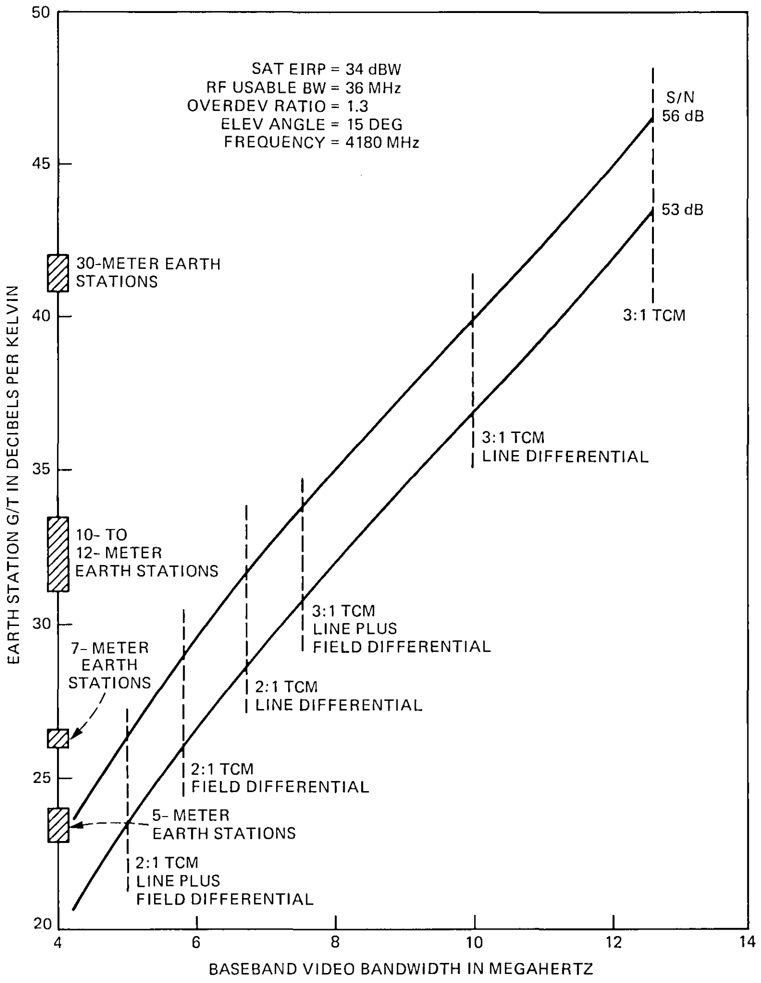


Fig. 7—Performance of a color TV transmission via TCM in a satellite link.

calculated by the well-known formula

$$\mathrm{s/n} = \mathrm{CNR} + 10 \log \left[12 \frac{(0.7\Delta f)^2 B}{f_m^3} \right] + 12.8, \tag{8}$$

where

CNR = received IF carrier-to-noise ratio (dB);

Δf = peak frequency deviation (MHz);

B = IF bandwidth (MHz);

f_m = baseband video bandwidth (MHz).

When Carson's rule is used, $B = 2(\Delta f + f_m)$. However, we assume that overdeviation* is permissible, and $\Delta f = 1.3(B/2 - f_m)$.

(*v*) The baseband bandwidth, f_m, would be equal to the bandwidth of an ordinary TV (i.e., f_t = 4.2 MHz) if it were a single TV/FM case. However, we let f_m vary over the range from 4.2 MHz to 12.6 MHz in the calculation to account for TCM.[4] The corresponding values of receive earth station figure of merit (G/T) for signal-to-noise ratio (s/n) = 53 and 56 dB are evaluated and plotted in the figure.

Referring to Fig. 7, we interpret the baseband video bandwidth f_m as the total baseband bandwidth of the combined time-compressed TV signals. For instance, the baseband bandwidth of a time-compressed TV coded with a line differential signal is 3.34 MHz (Section III). Time-compression multiplexing three such TV signals yields a total baseband bandwidth of 10.02 MHz. Using f_m = 10.02 MHz, we read from Fig. 7 that the G/T required to receive these three TV's at s/n = 56 dB is 39.8 dB. We recognize that this kind of calculation is not strictly valid because of the unknown subjective noise effects on the differential signals. However, these calculations do indicate the approximate performance to be expected and are suitable for the present feasibility study.

Along the vertical axis of Fig. 7, we also show some typical G/T's for differential types of earth stations. For easy reading we tabulate the baseband bandwidths for various cases of interest in Table I. Using Table I and Fig. 7, we derive various G/T's for the various cases, and the results for s/n = 56 and 53 dB are tabulated in Table II. If we assume that s/n = 53 dB is acceptable,[5] then the following observations can be drawn:

(*i*) Using 30-meter receive earth stations, we should be able to get

* Overdeviation with respect to the receiver noise bandwidth is sometimes used in satellite TV transmission to trade off waveform distortion for signal-to-noise performance. In this case, the modulator output would have to be filtered to ensure that the allocated bandwidth is not exceeded. In addition, the overall effect of the small amount of overdeviation used would have to be evaluated further.

Table I—Total baseband bandwidth for various combinations of bandwidth compression methods and TCM

	Total Baseband Bandwidth (MHz)		
Bandwidth Compression	1 TV	2 TV/TCM	3 TV/TCM
No Compression	4.2	8.4	12.6
Line Differential	3.34	6.68	10.02
Field Differential	2.9	5.8	8.7
Line and Field Differential	2.5	5.0	7.5

Table II—Receive earth station G/T required for various combinations of bandwidth compression methods and TCM

		G/T(dB/K)	
s/n	Bandwidth Compression Method	2 TV/TCM	3 TV/TCM
56	No Compression	35.9	46.4
	Line Differential	31.5	39.8
	Field Differential	28.9	36.7
	Line plus Field Differential	26.4	33.7
53	No Compression	32.9	43.4
	Line Differential	28.5	36.8
	Field Differential	25.9	33.7
	Line plus Field Differential	23.4	30.7

three TV's per transponder with the line differential compression method.

(*ii*) Using conventional 12-meter earth stations, or 10-meter earth stations with very sensitive low-noise amplifiers (LNAs) we should be able to get three TV's per transponder with the field or line plus field differential compression method, and two TV's per transponder with a straightforward TCM without baseband TV bandwidth reduction.

(*iii*) Using conventional 10-meter earth stations, we should be able to get two TV's per transponder with the line differential method.

(*iv*) Using 7-meter earth stations, we should be able to get two TV's per transponder with the field differential method.

(*v*) Using 5-meter earth stations, we should be able to get two TV's per transponder with the line plus field differential technique.

It is obvious that the above five results involve progressing bandwidth efficiency at the expense of increasing bandwidth reduction hardware.

The problem of adjacent satellite interference was not included in the above, and if the satellite spacing were to be reduced to 2° in the future, then the use of the 5- and 7-meter earth stations might not be appropriate. Finally, let us note that although the transmission of TV audios was not addressed, it could be included easily by virtue of the time-multiplexing characteristic in the system and the ease of digitizing audios today. A number of possibilities exist. Just to name one example, each TV line could be time compressed a little more to create sufficient time space for accommodating the digital audios.

V. CONCLUDING REMARKS

We have outlined three bandwidth compression techniques that can reduce the bandwidth of a 4.2-MHz NTSC color TV signal into 3.34, 2.9, and 2.5 MHz, respectively. These methods can be implemented with existing high-speed digital processing hardware. When these techniques are used in combination with TCM for multiple TV transmissions through a single satellite transponder, high bandwidth efficiency can be attained without the conventional problems owing to transponder nonlinearities (such as intermodulation and intelligible cross talk). We show in a specific example that up to three TV signals may be transmitted simultaneously with acceptable quality in a 36-MHz transponder with a 10-meter receive earth station, and with simpler hardware, two TV's per transponder may be transmitted with 7- or 10-meter earth stations. We conclude that these bandwidth compression techniques plus TCM are promising approaches to maximize the use of future satellite transponders for high-quality TV distribution.

REFERENCES

1. J. E. Flood and D. I. Urquhart-Pullen, "Time-Compression-Multiplex Transmission," Proc. IEE, *111,* No. 4 (April 1964), pp. 647–68.
2. D. H. Morgen and E. N. Protonotarios, "Time Compression Multiplexing for Loop Transmission of Speech Signals," IEEE Trans. on Commun., *COM-22,* No. 12 (December 1974), pp. 1932–9.
3. B. G. Haskell, "Time-Frequency Multiplexing of Two NTSC Color TV Signals—Simulation Results," B.S.T.J., *60,* No. 5 (May–June 1981), pp. 643–60.
4. K. Y. Eng and O. Yue, "Spectral Properties and Band-Limiting Effects of Time-Compressed TV Signals in a Time-Compression Multiplexing System," B.S.T.J., *60,* No. 9 (November 1981), pp. 2167–85.
5. *NTC Report No. 7,* The Public Broadcasting Service, Washington, D.C., June 1975, revised January 1976.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Sensitivity of Avalanche Photodetector Receivers for Long-Wavelength Optical Communications

By R. G. SMITH and S. R. FORREST

(Manuscript received May 27, 1982)

We consider, in detail, the potential improvement in receiver sensitivity that can be realized using an avalanche photodiode (APD) rather than the conventional p-i-n diode in long-wavelength optical communications systems. Numerical computations are used to determine optimum gains and receiver sensitivities for several values of ionization coefficient ratios and dark currents. Sensitivities are considered for transmission bit rates of 45 Mb/s, 90 Mb/s, and 274 Mb/s—values characteristic of present long-wavelength systems. We find that general relationships and scaling laws between receiver sensitivity and the other critical parameters can be formulated if the sensitivity is calculated in units relative to the quantum limit. An important result is that the improvement in APD sensitivity depends strongly on dark current, but only weakly on the ionization coefficient ratio. Our calculations are compared with recent results obtained for $In_{0.53}Ga_{0.47}As/InP$ APDs sensitive in the λ = 0.95 μm to 1.6 μm wavelength region. We also include a brief discussion comparing APD sensitivities with those obtained using phototransistors and majority carrier devices.

I. INTRODUCTION

In contrast to short wavelength (0.8 μm to 0.9 μm) lightwave transmission systems, which generally use an avalanche photodiode (APD) as the detection element,[1] many receivers for long wavelength (1.3 μm to 1.55 μm) lightwave systems use p-i-n detectors.[2,3] The principal reason for using the APD at short wavelengths is the ~15-dB improvement in sensitivity obtained compared with a p-i-n with a nonintegrating front end. The arguments given for using p-i-ns at long wavelengths are: (*i*) sensitivities comparable to those obtained at short wavelengths can be achieved using GaAs field-effect transistor (FET) front-end amplifiers; (*ii*) the ionization coefficients for electrons and holes in InP and related InGaAsP compounds are not significantly different, leading to large excess noise factors and, hence, poor receiver sensitivities; and (*iii*) the APD is difficult to fabricate, especially with low dark currents.

In this paper the subject of receiver sensitivity using APDs is addressed with emphasis placed on the limits imposed by the ionization coefficients and the device dark currents. The principal result is that whereas the ratio of the ionization coefficients determines the maximum sensitivity improvement attainable with an APD, the degree to which this improvement can be achieved in practice is controlled by the dark current. It is concluded that, for sufficiently low dark currents, an APD can yield significant sensitivity improvements over that obtained using a p-i-n detector, resulting in greater regenerator spacings, or permitting the use of less sophisticated amplifiers with greater dynamic range and lower cost.

1.1 Detector alternatives

In choosing a detector for lightwave receivers there are several alternatives to the p-i-n and APD, including the phototransistor,[4,5] majority carrier devices,[6] and photoconductors.[7,8] It has been claimed that these devices can produce gains comparable to, or in excess of, those attained with an APD but without the penalty of an excess noise factor. It is not, however, the value of an excess noise factor that is important, but rather the total noise contributed by the device in producing the associated gain. In contrast to the p-i-n and APD, which achieve high-speed operation under reverse bias, the phototransistor, the majority carrier detector, and the photoconductor all draw current when biased to obtain high-speed operation. These bias currents contribute significant noise to the receiver and limit the sensitivity. For the APD, dark currents also affect the sensitivity obtained but they are not, however, a necessary part of the high-speed operation, and can in principle be reduced to sufficiently low levels to achieve significant receiver sensitivity improvements.

1.2 Scope and organization

In the material to follow, APD receiver sensitivities will be characterized with respect to bit rate, amplifier noise, ratio of ionization coefficients, and the detector dark current. The sensitivity improvement achieved as a function of the above parameters will be calculated and compared with recently reported results. The paper is organized as follows: in Section II we consider the base-line receiver sensitivity and develop the formalism for calculating sensitivity improvements afforded with an APD. In Section III, we calculate the improvement obtained for an APD with zero dark current, and in Section IV we consider the effects of the dark current on degradation of the receiver sensitivity. In Section V, we present a general scaling rule for APD receiver sensitivity, and in Section VI we present sample calculations of receiver sensitivities of current interest, and compare the results of a recent measurement of a long-wavelength APD receiver sensitivity with our calculations. Finally, in Section VII, we present conclusions.

II. RECEIVER SENSITIVITY

In analyzing the sensitivity of a digital receiver, we know that for a given decision level the net signal is a factor Q times the root-mean-square (rms) noise associated with the state transmitted[9] (mark or space), with $Q = 6$ for a bit error rate (BER) of 10^{-9}. For practical p-i-n detectors the noise is determined by the amplifier and the detector leakage currents, and is identical for both signal states. In this case the decision level is placed midway between the two signal levels and the sensitivity is given by

$$\eta\bar{P} = A\langle i^2\rangle^{1/2}, \tag{1}$$

where η is the detector quantum efficiency including coupling losses, $\bar{P}$ is the average optical power required to achieve a given BER, and $\langle i^2\rangle^{1/2}$ is the rms noise current of the receiver referred to the input. The parameter A is given by

$$A = Q\frac{h\nu}{q}, \tag{2}$$

where $h\nu$ is the energy of the incident radiation and q is the electronic charge. For a BER $= 10^{-9}$, $A = 9$ at $\lambda = 0.825$ μm, and $A = 5.7$ at $\lambda = 1.3$ μm. Whereas the noise current, $\langle i^2\rangle$, is normally used to characterize receivers, in the following calculations the total receiver sensitivity using a p-i-n detector will be used instead. The relation between the two quantities is given by eq. (1).

When an APD is used, both the signal current and the shot-noise associated with the signal current are multiplied by the avalanche process. For avalanche gains near the optimum value, the shot-noise associated with the signal is comparable to the receiver noise. Under these conditions the decision level, or threshold, is no longer midway between the two signal levels but is located closer to the space, or "0" level. Under the assumption that the received signal power is essentially zero in the off state (perfect source extinction), and that the detector dark current that undergoes multiplication is also zero, the receiver sensitivity is given by[10]

$$\eta \bar{P} = A \left[\frac{\langle i^2 \rangle^{1/2}}{M} + (qBI_1)QF(M) \right], \tag{3}$$

where M is the average gain, B is the bit rate, I_1 is a Personick integral ≈ 0.5, and $F(M)$ is the excess noise factor associated with the avalanche gain process. In deriving eq. (3), the optimum decision level has been assumed for each value of the gain M. In the limit $M = 1$, and neglecting the second term in brackets, which is small, eq. (3) is seen to reduce to eq. (1). In evaluating eq. (3) we use McIntyre's expression[11] for the excess noise factor, $F(M)$:

$$F(M) = M \left[1 - (1 - k) \left(\frac{M - 1}{M} \right)^2 \right]. \tag{4}$$

Here, $k \leq 1$ is the ratio of the ionization coefficients of the ionizing carriers (holes or electrons) where it is assumed the avalanche is initiated by the carrier with the highest ionization rate.

Under more general circumstances of receiver operation the current produced by the optical source is not identically zero in the "0" level, nor is the detector dark current which undergoes multiplication necessarily negligible. Each source of current undergoing avalanche gain produces shot-noise that contributes to the total noise in both the "0" and "1" states. In general, the primary current should be less than approximately 10 percent of the photocurrent associated with the "1" state if its effect on sensitivity is to be small.

The effects of detector dark current can be handled by including both the unmultiplied and multiplied components as part of the receiver noise. Thus, the receiver noise becomes

$$\langle i^2 \rangle_{\text{total}} = \langle i^2 \rangle_a + 2qI_2B[I_{DU} + I_{DM}M^2F(M)], \tag{5}$$

where $\langle i^2 \rangle_a$ is the amplifier noise independent of the detector leakage current, I_{DU} is the unmultiplied portion of the dark current, I_{DM} is the primary dark current, which undergoes avalanche gain, and I_2 is a Personick integral,[9,10] which typically has a value between 0.5 and 0.6. In using eq. (5), it is assumed that the noise is Gaussian; also in the following treatment we assume that the detector dark current undergoes the same gain and has the same excess noise factor as the photogenerated current.

It can be shown that when detector dark current is considered, eq. (3) can be used to evaluate the sensitivity if $\langle i^2 \rangle$ in eq. (3) is replaced with $\langle i^2 \rangle_{\text{total}}$, eq. (5). In this case the effective receiver noise is a function of the gain, M, with the result that simple analytical optimization of eq. (3) is difficult. In the calculations to follow, eq. (3), with the modification discussed above, has been numerically evaluated to find an optimum value of the gain. The results are obtained with the assumption of complete source extinction. The effect of a nonideal source can be obtained by finding the equivalent extinction ratio corresponding to the assumed dark current and the primary signal current. Note also that, from eq. (5), the effect of unmultiplied dark currents such as surface leakage, I_{DU}, is in general not significant compared with the multiplied dark currents, I_{DM}. For example, in a receiver with an optimum gain of $M = 10$, $I_{DM} = 1$ nA contributes more to the total receiver noise than $I_{DU} = 100$ nA (here $F(M) \lesssim M$). Thus, in the remainder of this work we focus on the multiplied portion of the dark current, including I_{DU} in the basic amplifier noise.

III. SENSITIVITY CALCULATIONS ($I_{DM} = 0$)

The sensitivity calculations to be presented here are given in detail for several bit rates of current interest. By normalizing sensitivities to the quantum limit it is shown that a universal sensitivity curve independent of bit rate can be derived. The parameters to be considered are the amplifier noise characterized by the equivalent receiver sensitivity, the effective ratio of ionization coefficients, k, and the primary dark current, I_{DM}, which undergoes multiplication.

Figure 1 is a plot of sensitivity at $\lambda = 1.3$ μm and 45 Mb/s for four values of k in the limit that the primary dark current is zero (perfect extinction of the source is assumed). The calculated sensitivities thus represent the best attainable values. Here $\eta \bar{P}_{PIN}$ denotes the sensitivity of the amplifier using a p-i-n detector. Similarly, the vertical axis is the sensitivity of the same amplifier employing an APD. The numerical values in parentheses along each curve represent the optimum gain for the corresponding set of parameters. For example, in Fig. 1 a p-i-n receiver with a −50 dBm sensitivity would have a sensitivity of −56.5 dBm with an APD having $k = 1$ at an optimum gain of $M_{\text{opt}} \cong 9$, and for $k = 0.025$ the sensitivity improves to −62.3 dBm at $M_{\text{opt}} \cong 60$. The p-i-n sensitivity of −50 dBm is typical of that achievable with GaAs FET front ends,[3] while the assumed k values are typical[12] of Ge ($k = 1$) and the best Si devices ($k \cong 0.025$).

Several general features may be seen from these curves: Decreasing k results in a larger sensitivity improvement, and over the range of p-i-n sensitivities shown, the improvement achieved by the lowest k-value device is between 5 dB and 7 dB greater than the $k = 1$ device. On the other hand, the improvement relative to a p-i-n afforded by the $k = 1$ device is between 4.5 and 11.5 dB. Also, as the p-i-n receiver sensitivity increases, the improvement achieved with an APD diminishes. For $k = 1$, a 2-dB improvement in p-i-n sensitivity results in a 1-dB improvement for the same amplifier when using an APD. However, for $k = 0.025$, an improvement in p-i-n sensitivity of 3 dB is required to achieve the same 1-dB improvement when using an APD. Thus, the

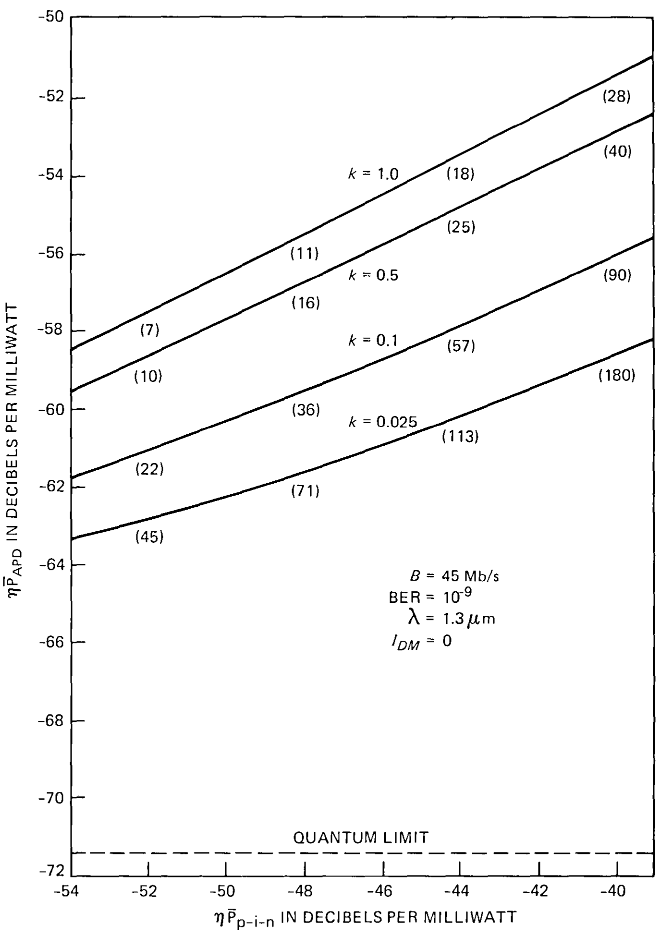


Fig. 1—APD sensitivity versus p-i-n sensitivity at a bit rate of 45 Mb/s and zero dark current (I_{DM}), for several values of the ionization coefficient ratio, k. Numbers in parentheses indicate optimum gains yielding the corresponding value of APD sensitivity.

improvement to be gained with an APD diminishes as the amplifier performance is improved; conversely, the APD is more tolerant of poorer amplifier noise performance. Hence, an APD can permit the use of higher noise amplifiers, which in many cases, possess greater dynamic range. Another characteristic of APD receivers seen from Fig. 1 is that the optimum gain (M_{opt}) decreases as k increases, and also M_{opt} decreases as the p-i-n sensitivity improves. For the range of parameters shown in the figure, the improvement afforded by an APD is between $M_{\text{opt}}/2$ and $M_{\text{opt}}/3$.

Curves of sensitivity for bit rates of 90 Mb/s and 274 Mb/s are shown in Figs. 2 and 3, respectively. The range of p-i-n and APD

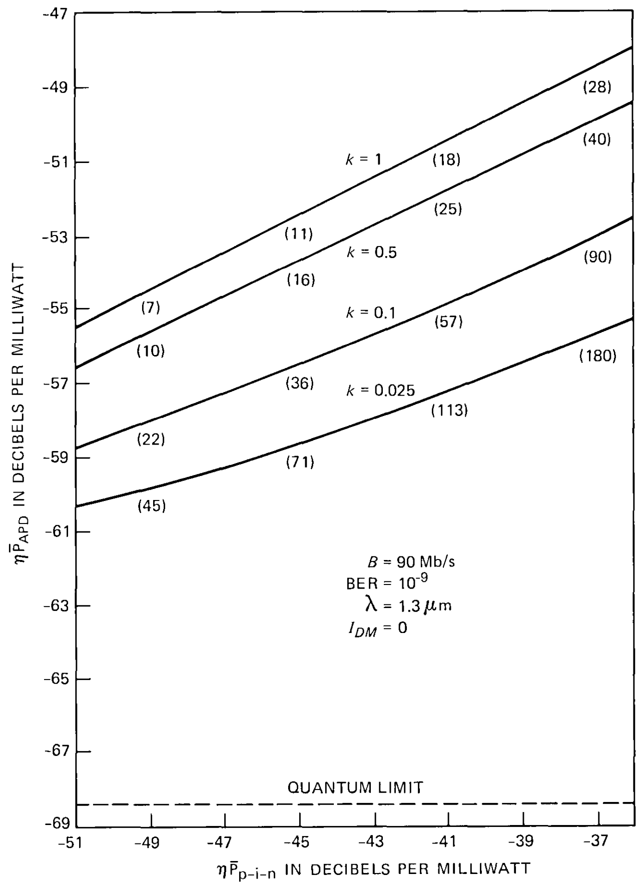


Fig. 2—Similar to Fig. 1 with B = 90 Mb/s.

sensitivities covered by these curves are adjusted to include values that might be achieved at these different bit rates. Extrapolation to other bit rates can be made by noting that Figs. 1 to 3 are essentially identical except that the horizontal and vertical axes are scaled by a constant amount, dependent upon the bit rate. For example, Fig. 2 (for 90 Mb/s) can be derived from Fig. 1 (for 45 Mb/s) by adding 3 dB to both the ordinate and abscissa, e.g., −50 dBm at 45 Mb/s becomes −47 dBm at 90 Mb/s, etc. This simple multiplicative scaling rule follows from eq. (3) and the method of defining the receiver noise in terms of the p-i-n sensitivity.

A curve applicable to all bit rates can be generated by defining all sensitivity values relative to the quantum limit, $\eta\bar{P}_{QL}$, which for a bit error rate of 10^{-9}, is given by,

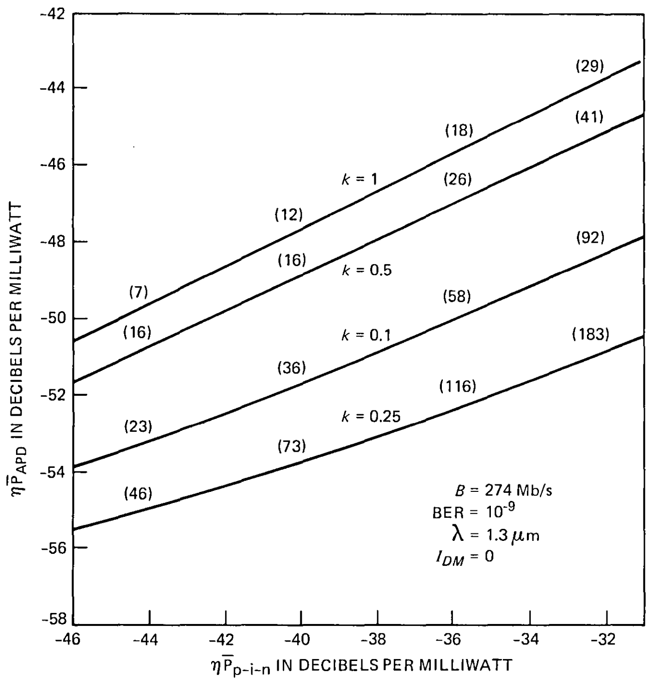


Fig. 3—Similar to Fig. 1 with B = 274 Mb/s.

$$\eta \bar{P}_{QL} = \frac{21}{2}\,(h\nu)B. \tag{6}$$

This expression corresponds to 21 transmitted photons per mark, and a mark-to-space ratio of one. For $\lambda = 1.3\ \mu$m, eq. (6) becomes

$$\eta \bar{P}_{QL}\ (\text{dBm}) = -87.95 + 10 \log_{10} B\ (\text{Mb/s}). \tag{7}$$

Values of quantum limited sensitivity are plotted in Fig. 4.

Figure 5 is a universal plot of the improvement in sensitivity achieved with an APD with zero multiplied dark current and independent of bit rate for several values of k. The vertical axis is the improvement in sensitivity (in dB) achieved with an APD compared to a p-i-n using the same amplifier, while the horizontal axis is the difference between the sensitivity of the p-i-n receiver and the quantum limit. Again, the advantage provided by an APD is seen to diminish as the p-i-n sensitivity approaches the quantum limit. As an example, the current receiver designs using p-i-n/GaAs FET receivers have sensitivities typically in the range of 20 to 25 dB above the quantum limit.[2,3] The maximum sensitivity improvements afforded by an APD are thus greater than 6 dB for this range of p-i-n sensitivities, depending on the k-value.

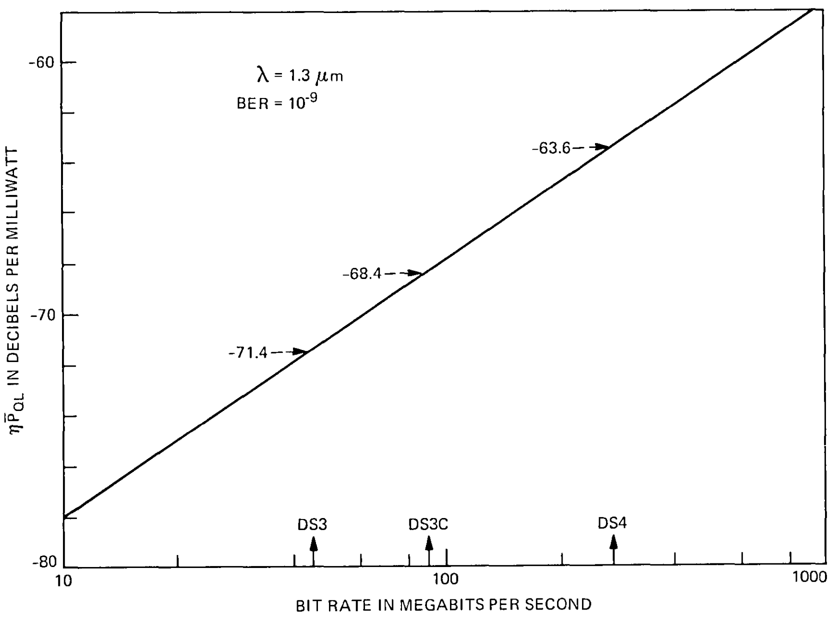


Fig. 4—Quantum limit ($\eta\bar{P}_{QL}$) versus bit rate. Numbers indicate values for standard transmission system bit rates as labeled on the abscissa.

IV. SENSITIVITY CALCULATIONS ($I_{DM} > 0$)

The sensitivity improvements predicted above correspond to an ideal source and detector, in that the source extinction ratio is infinite and the dark current is negligible. In this section we investigate how much this improved performance is degraded by detector dark current. Since the absolute sensitivity depends upon dark current, the ratio of ionization coefficients, the receiver noise as characterized by the p-i-n sensitivity, and the bit rate, a large number of curves would have to be presented to discuss the full range of operating possibilities. Figures 6, 7, and 8 show the effect of primary dark current on sensitivity for bit rates of 45 Mb/s, 90 Mb/s, and 274 Mb/s. The p-i-n sensitivities indicated on each curve correspond to values currently achievable with optimized GaAs FET amplifiers. The arrows on the left-hand axis indicate the limiting sensitivity ($I_{DM} = 0$) calculated in Section III above. The general features observed in these figures are the falloff of the sensitivity with increasing dark current, and the increased sensitivity to dark current as k decreases.

For example, at 45 Mb/s (Fig. 6), $I_{DM} = 10^{-10}$ A reduces the maximum obtainable improvement for $k = 0.025$ by ≈ 1 dB, while it has little impact for $k = 1$. At 90 Mb/s (Fig. 7) the same degradation for $k = 0.025$ occurs for $I_{DM} \approx 2 \times 10^{-10}$ A, and at 274 Mb/s (Fig. 8) it occurs for $I_{DM} \approx 7 \times 10^{-10}$ A. Thus, at higher bit rates, the APD is tolerant of

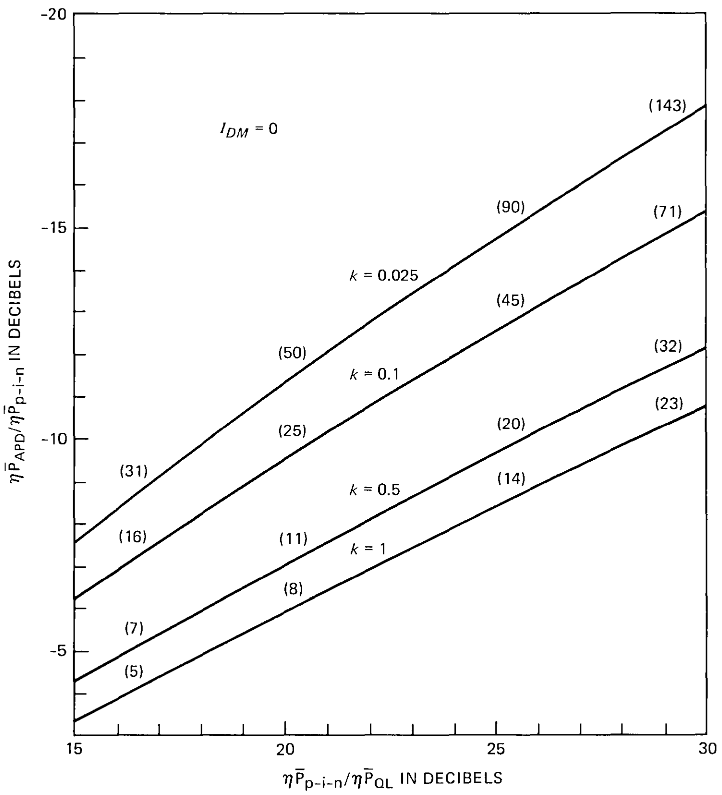


Fig. 5—Sensitivity improvement of the APD over p-i-n sensitivity versus normalized p-i-n sensitivity for zero dark current (I_{DM}), and for several values of the ionization coefficient ratio, k. Numbers in parentheses indicate optimum APD gain. This plot is independent of bit rate.

greater dark current levels for equal sensitivity degradation. As we discuss below, the dark current producing a given degradation is directly proportional to the bit rate. A second feature of Figs. 6, 7, and 8 is that, whereas a lower k-value produces an improvement in sensitivity, the amount of improvement decreases with increasing primary dark current.

In a manner similar to that used to generate Fig. 5, a general curve, independent of bit rate, can be generated showing the effect of dark current on sensitivity. In Fig. 9 the p-i-n sensitivity is assumed to be 21.4 dB above the quantum limit, and the dark current normalized to the bit rate is expressed in nA/(Mb/s). The APD sensitivity relative to the quantum limit is shown on the left-hand axis, and the improvement afforded by the APD compared to the p-i-n is shown on the right-hand axis. Again, the rapid decrease in the advantage of an APD with increasing dark current is evident. Further, the advantage gained

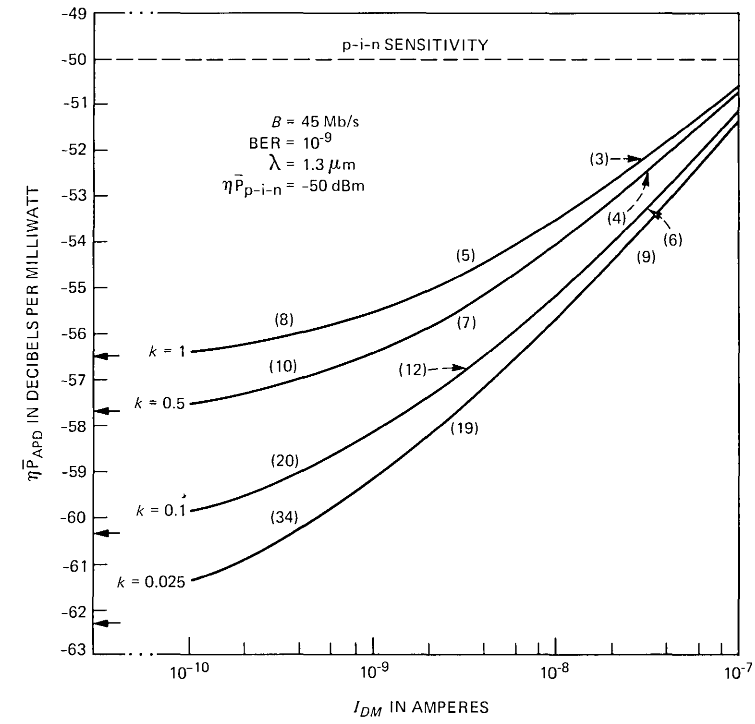


Fig. 6—APD sensitivity at 45 Mb/s versus primary dark current (I_{DM}) for several values of the ionization coefficient ratio, k. Arrows along the vertical axis correspond to the APD sensitivity at $I_{DM} = 0$.

by using a low k-value APD compared to one with a larger k value is seen to be significant only when the dark currents are low, i.e., less than 10^{-2} nA/Mb/s.

The above calculations assume a p-i-n sensitivity very nearly equal to the best presently achievable. Figure 10 shows the effect of varying the p-i-n sensitivity. The figure is plotted for $k = 0.5$, a value typical of present $In_{0.53}Ga_{0.47}As/InP$ APDs.[13,14] The left-hand and lower axes are plotted in normalized units, whereas the right-hand and upper axes are shown unnormalized for the three bit rates considered. The p-i-n sensitivities for each of the bit rates are shown in the box at the left. Included are values 3 dB better, and 3 dB and 6 dB worse than those used in the preceding calculations. From these curves we see that for low dark current values an improvement in p-i-n sensitivity by 3 dB results in a 1.4-dB improvement in APD sensitivity, whereas this value is decreased to $\approx$1 dB for the higher range of dark currents, i.e., 1 nA/(Mb/s). Similar curves can be generated for other values of k but will not be presented here.

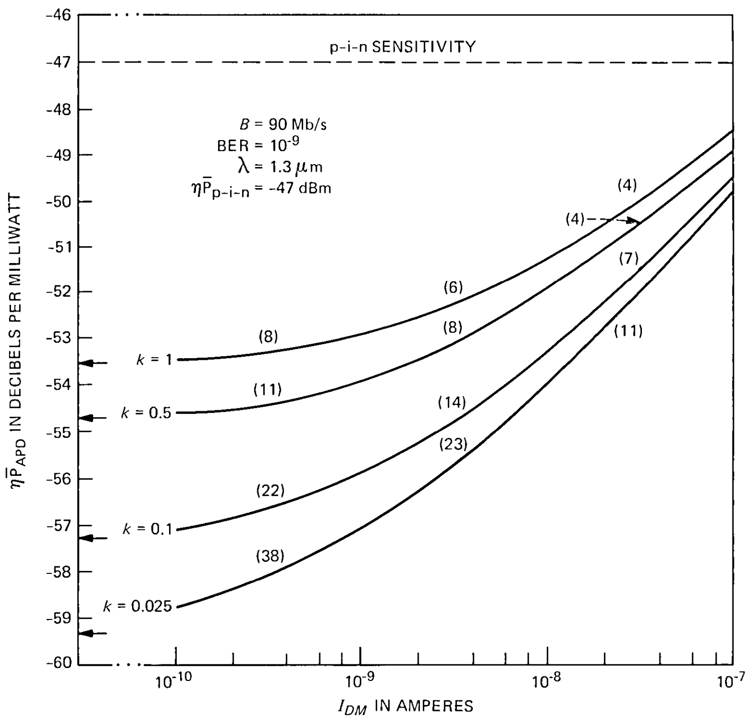


Fig. 7—Similar to Fig. 6 with B = 90 Mb/s.

V. GENERALIZED RELATION FOR ALLOWABLE DARK CURRENT

Because of the number of parameters involved in determining sensitivity, the preceding calculations have assumed a particular value for either the dark current, k value or p-i-n sensitivity. We now derive a relation between these quantities that defines the maximum allowable dark current for a given degradation of the sensitivity from the ideal ($I_{DM} = 0$) case.

We begin by combining eqs. (1) to (3), (5), and (6) to give

$$\frac{\eta \bar{P}_{APD}}{\eta \bar{P}_{QL}} = \frac{\left[\left(\frac{\eta \bar{P}_{p\text{-}i\text{-}n}}{\eta \bar{P}_{QL}}\right)^2 + \left(\frac{4}{7}\right)^2 \frac{I_{DM}}{qB} M^2 F\right]^{1/2}}{M} + \frac{12}{7} F. \qquad (8)$$

In obtaining the numerical values in eq. (8), the Personick integrals I_1 and I_2 have both been set equal to 0.5, and we have used $Q = 6$. The unmultiplied dark current, I_{DU}, is included in the p-i-n sensitivity. This expression is independent of bit rate provided I_{DM}/B is constant. Minimizing eq. (8) with respect to M in the limit $I_{DM} = 0$, and using eq. (4) for F gives the results presented in Fig. 5. Defining

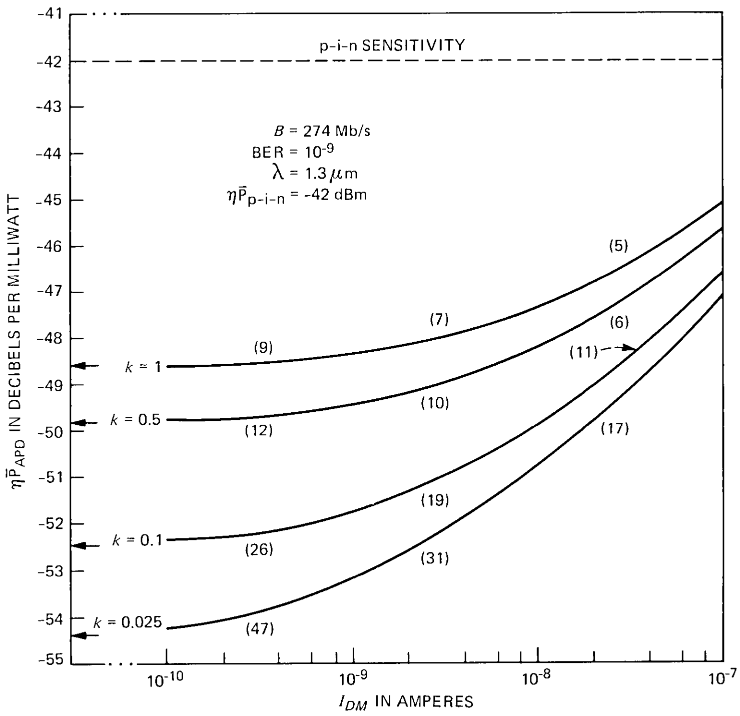


Fig. 8—Similar to Fig. 6 with B = 274 Mb/s.

$$R = \frac{\eta \bar{P}_{p\text{-}i\text{-}n}}{\eta \bar{P}_{QL}}, \tag{9}$$

the optimum gain, M_{opt}, for $I_{DM} = 0$ is then given by

$$M_{\text{opt}} = \left(\frac{7}{12} R + k - 1\right)^{1/2} \bigg/ k^{1/2}. \tag{10}$$

In this limit eqs. (8) to (10) yield

$$\left(\frac{\eta \bar{P}_{APD}}{\eta \bar{P}_{QL}}\right)_{I_{DM}=0} = \frac{24}{7} (kM_{\text{opt}} + 1 - k). \tag{11}$$

If the APD sensitivity is now degraded by some amount, for example by 1 dB because of non-zero dark current, eqs. (8) and (11) can be combined to define the locus of gains, M, and normalized dark currents, I_{DM}/qB, yielding that degradation. The maximum value of I_{DM}/qB for which a positive real value of M exists then defines the maximum value of the normalized dark current for which the degraded sensitivity can be obtained. Figure 11 shows a plot of the normalized dark current that results in a 1-dB degradation from the ultimate attainable im-

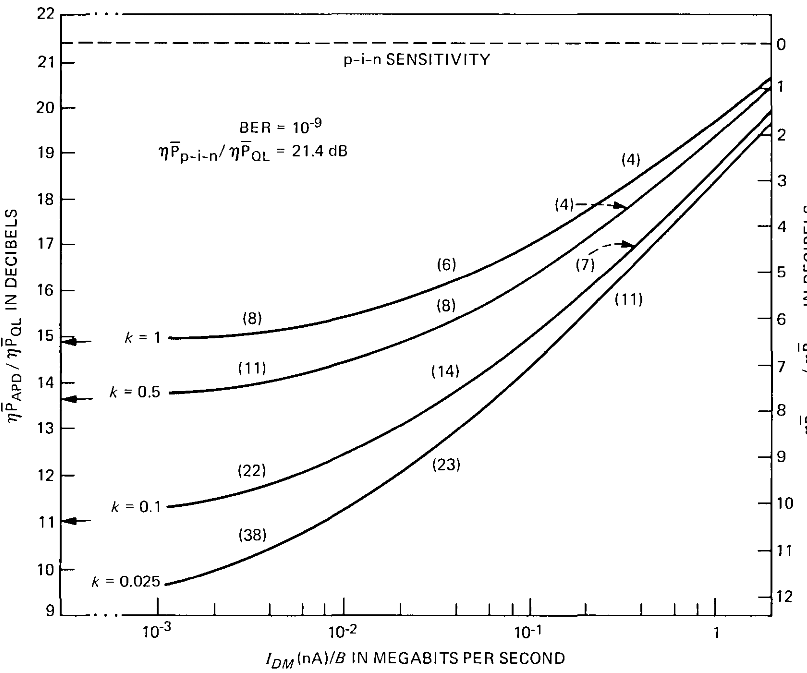


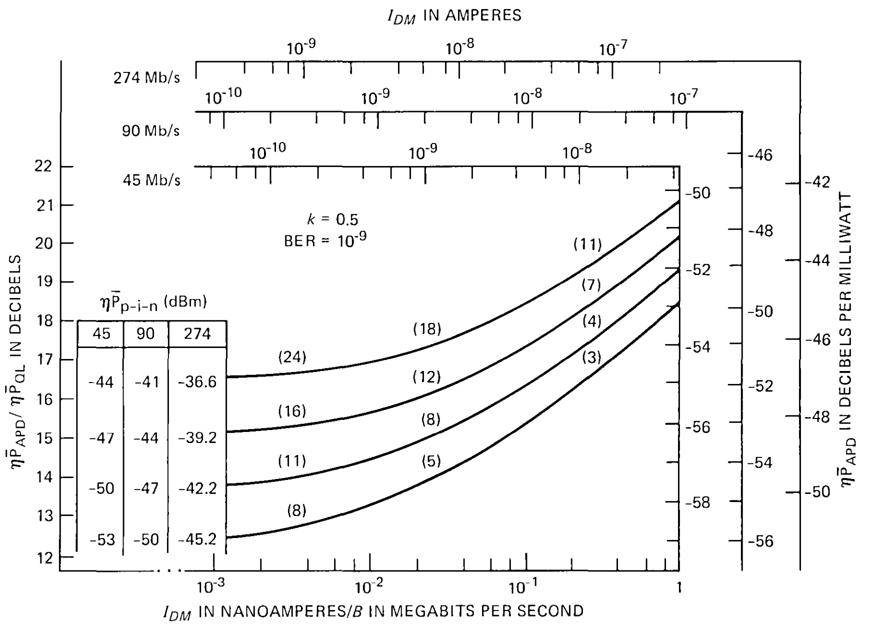
Fig. 9—APD sensitivity referred to the quantum limit versus primary dark current (I_{DM}) per bit rate (B) for several values of the ionization coefficient ratio, k. Numbers in parentheses indicate the optimum gains needed to achieve the corresponding APD sensitivities. Arrows along the vertical axis correspond to the APD sensitivity obtained at $I_{DM} = 0$.

provement ($I_{DM} = 0$). As an example, a receiver with a p-i-n sensitivity 20 dB above the quantum limit and with $k = 0.5$ will suffer a 1-dB degradation at $I_{DM}/qB \cong 1.4 \times 10^{-2}$ nA/Mb/s. At a bit rate of 45 Mb/s, this corresponds to a primary dark current of ≈0.6 nA. On the other hand, at 274 Mb/s the same degradation would result for $I_{DM} \approx$ 4 nA. For a Si detector with $k \approx 0.025$, the primary dark currents yielding a 1-dB degradation would be a factor of six lower in each case. From Fig. 11 we see that the maximum permissible dark current decreases as the p-i-n receiver sensitivity increases and also as the k-value is decreased. Since the curves in Fig. 11 are nearly straight and parallel lines, they can be fit by a simple, approximate relation accurate to better than 20 percent over practical values of R and k. Therefore,

$$I_{DM}(\text{nA}) \approx 2 \times 10^{-3}(Rk)^{1/2}B \text{ (Mb/s)} \tag{12}$$

for a 1-dB penalty. Thus, an improvement in p-i-n sensitivity by a factor of 4, or a decrease of k by a factor of 4, would result in a reduction in the maximum allowable dark current by a factor of 2.

The above relation and Fig. 11 are for a degradation in sensitivity of



45	90	274
-44	-41	-36.6
-47	-44	-39.2
-50	-47	-42.2
-53	-50	-45.2

Fig. 10—Similar to Fig. 9 with $k = 0.5$ for several values of p-i-n receiver sensitivity ($\eta\bar{P}_{p\text{-}i\text{-}n}$) and bit rate.

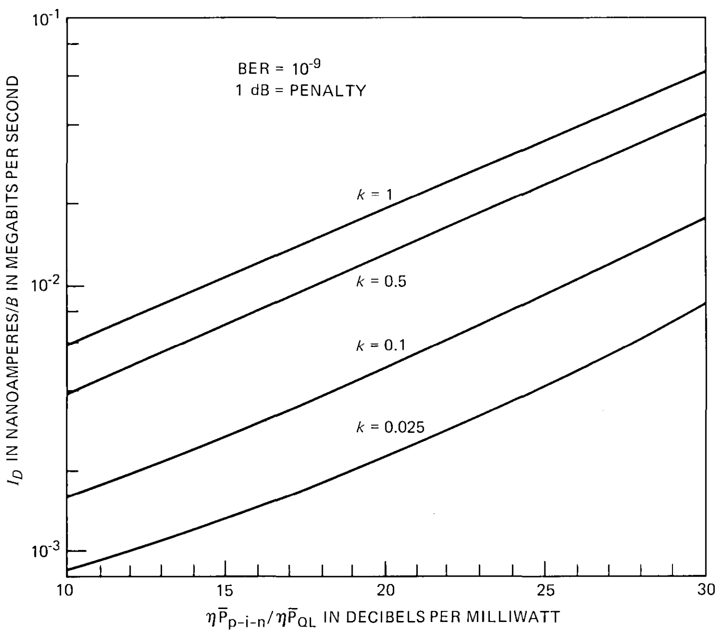


Fig. 11—Allowable primary APD dark current (I_{DM}) per bit rate (B) resulting in a 1-dB penalty from ultimately attainable APD sensitivity, as a function of p-i-n receiver sensitivity normalized to the quantum limit. Plots for several values of the ionization coefficient ratio, k, are shown.

1 dB. For a degradation in sensitivity of 2 dB, the corresponding primary dark current values are increased by a factor of 3.4.

VI. EXAMPLES

Using the above results, we calculate the expected sensitivity improvements that might be obtained and the dark current limitations for several possible bit rates. We consider a p-i-n receiver yielding a sensitivity of 20 dB above the quantum limit (−51.4 dBm at 45 Mb/s, −48.4 dBm at 90 Mb/s, and −43.6 dBm at 274 Mb(s). This value lies between values achieved to date and those achievable assuming circuit improvements. We consider both $k = 0.5$ and 0.1 values typical of present devices,[13,14] as well as more sophisticated structures.[15]

Table I gives the maximum permissible dark current as a function of bit rate at which point the sensitivity values are degraded by 2 dB from the ideal, i.e., 5-dB net improvement for $k = 0.5$, and 7.5-dB net improvement for $k = 0.1$ (see Fig. 5). As we see in Table I, the permissible current values increase with bit rate, and decrease by a factor ≈ 2 for a reduction by a factor of 5 in k. Whereas the values in Table I are comparable to dark current values presently attainable,[13,16] it is noted that these are maximum values, and hence correspond to dark currents obtained at the highest operating temperature of the device. For a maximum temperature of 70°C typical of most system requirements, the increase in dark current compared to room temperature is typically a factor of 8.[17] Hence, if the detector is to operate under the assumed conditions, the room temperature dark currents must be a factor of 8 below the values shown in Table I, i.e., 250 pA for a 45-Mb/s system and $k = 0.5$.

We now compare our calculations to recent measurements of sensitivity made on a long-wavelength $In_{0.53}Ga_{0.47}As/InP$ APD receiver. In the measurement of Forrest et al.,[13] at $B = 45$ Mb/s, an APD receiver sensitivity of $\eta \bar{P}_{APD} = -53.2$ dBm and at $k \simeq 0.5$ was reported. At this bit rate, Fig. 10 gives $I_{DM} = 2.5$ nA at the measurement temperature of 20°C. The diode area was $A = 1.3 \times 10^{-4}$ cm^2. For detectors presently being considered, a reduced area of $A \simeq 5 \times 10^{-5}$ cm^2 would in turn reduce the bulk current to $I_{DM} \cong 1$ nA at 20°C. Thus, at a maximum operating temperature of 70°C, we expect $I_{DM} \cong 8$ nA. In Table II we

Table I—Maximum primary dark current (nA) giving a 2-dB degradation in APD sensitivity

Bit Rate (Mb/s)	$k = 0.5$	$k = 0.1$
45	2	0.8
90	4	1.5
274	12	4.5

Table II—Improvement in receiver sensitivity (in dB) using a state-of-the-art APD with I_{DM} = 8 nA at 70°C and k = 0.5

	B(Mb/s)		
$\eta\bar{P}_{\text{pin}}$(dBm)	45	90	274
−50	4.3*	3.1	1.9
−47	6.3	5.1*	3.6
−42	9.6	8.6	6.4*

* Numbers refer to state-of-the-art receiver sensitivities at corresponding bit rates.

Table III—Improvement in receiver sensitivity using an APD with I_{DM} = 800 pA at 70°C and k = 0.5

	B(Mb/s)		
$\eta\bar{P}_{\text{pin}}$(dBm)	45	90	274
−50	6.5*	5.1	3.0
−47	8.5	7.1*	4.7
−42	11.5	9.7	7.5*

* Numbers refer to state-of-the-art receiver sensitivities at corresponding bit rates.

show the improvement in receiver sensitivity afforded by this APD at several values of bit rate and receiver sensitivities. As is evident, the improvement increases with bit rate and with increasing receiver noise. For comparison, in Table III we show the sensitivity improvement if the APD dark current were reduced by a factor of ten (i.e., 100 pA at 20°C or 800 pA at 70°C).

VII. CONCLUSIONS

We have calculated the improvement in sensitivity of receivers employing a long-wavelength avalanche photodiode rather than the traditionally used p-i-n detectors. The calculations consider receivers operating at bit rates of 45 Mb/s, 90 Mb/s, and 274 Mb/s, and sensitivities have been calculated for a wide range of I_{DM} and k values. We have found a strong degradation of receiver sensitivity with increasing APD dark current, with a much weaker dependence of sensitivity on the ionization coefficient ratio, k. Indeed, to fully realize the potential of APD's based on InP heterostructures, a considerable reduction in presently attainable dark currents is required.

REFERENCES

1. R. G. Smith, C. A. Brackett, and H. W. Reinbold, "Optical Detector Package," B.S.T.J., *57,* No. 6 (July–August 1978), pp. 1809–22.

2. D. R. Smith, R. C. Hooper, K. Ahmad, D. Jenkins, A. W. Mabbit, and R. Nicklin, "p-i-n/F.E.T. Hybrid Optical Receiver for Longer-Wavelength Optical Communication Systems," Electron. Lett., *16* (January 1980), pp. 69–71.
3. D. Gloge, A. Albanese, C. A. Burrus, E. L. Chinnock, J. A. Copeland, A. G. Dentai, T. P. Lee, Tingye Li, and K. Ogawa, "High-Speed Digital Lightwave Communication Using LEDs and PIN Photodiodes at 1.3 μm," B.S.T.J., *59,* No. 8 (October 1980), pp. 1365–82.
4. D. Fritzsche, E. Kuphal, and R. Aulbach, "Fast Response InP/InGaAsP Heterojunction Phototransistors," Electron. Lett., *17,* No. 5 (March 1981), pp. 178–80.
5. J. C. Campbell, C. A. Burrus, A. G. Dentai, and K. Ogawa, "Small-Area High-Speed InP/InGaAs Phototransistor," Appl. Phys. Lett., *39,* No. 10 (November 15, 1981), pp. 820–1.
6. C. Y. Chen, A. Y. Cho, P. A. Garbinski, C. G. Bethea, and B. F. Levine, "Modulated Barrier Photodiode: A New Majority-Carrier Photodetector," Appl. Phys. Lett., *39,* No. 4 (August 15, 1981), pp. 340–2.
7. J. Barnard, H. Ohno, C. E. C. Wood, and L. F. Eastman, "Integrated Double Heterostructure $Ga_{0.47}In_{0.53}As$ Photoreceiver With Automatic Gain Control," IEEE Electron. Dev. Lett., *EDL-2,* No. 1 (January 1981), pp. 7–9.
8. J. Degani, R. F. Leheny, R. E. Nahory, M. A. Pollack, J. P. Heritage, and J. C. DeWinter, "Fast Photoconductive Detector Using p-$In_{0.53}Ga_{0.47}As$ With Response to 1.7 μm," Appl. Phys. Lett., *38* (January 1981), pp. 27–9.
9. S. D. Personick, "Receiver Design for Digital Fiber Optic Communication Systems I," B.S.T.J., *52,* No. 6 (July–August 1973), pp. 843–74.
10. R. G. Smith and S. D. Personick, "Receiver Design for Optical Fiber Communication Systems," *Topics in Applied Physics, 39,* H. Kressel, Ed., Berlin: Springer-Verlag, 1979, pp. 89–160.
11. R. J. McIntyre, "Multiplication Noise in Uniform Avalanche Diodes," IEEE Trans., Electron. Dev. *ED-13* (January 1966), pp. 164–8.
12. G. E. Stillman and C. M. Wolfe, "Avalanche Photodiodes," *Semiconductors and Semimetals, 12,* Willardson and Beers, Ed., New York: Academic Press.
13. S. R. Forrest, G. F. Williams, O. K. Kim, and R. G. Smith, "Excess-Noise and Receiver Sensitivity Measurements of $In_{0.53}Ga_{0.47}As$/InP Avalanche Photodiodes," Electron. Lett., *17,* No. 24 (November 1981), pp. 917–9.
14. N. Susa, H. Nakagome, O. Mikami, H. Ando, and H. Kanbe, "New InGaAs/InP Avalanche Photodiodes Structure for the 1–1.6 μm Wavelength Region," IEEE J. Quant. Electron., *QE-16,* No. 8 (August 1980), pp. 864–9.
15. F. Capasso, W. T. Tsang, A. L. Hutchinson, and G. F. Williams, "Enhancement of Electron Impact Ionization in a Superlattice: A New Avalanche Photodiode With a Large Ionization Rate Ratio," Appl. Phys. Lett., *40,* No. 1 (January 1, 1982), pp. 38–40.
16. O. K. Kim, S. R. Forrest, W. A. Bonner, and R. G. Smith, "A High Gain $In_{0.53}Ga_{0.47}As$/InP Avalanche Photodiode With No Tunneling Leakage Current," Appl. Phys. Lett., *39* (September 1981), pp. 402–4.
17. S. R. Forrest, R. F. Leheny, R. E. Nahory, and M. A. Pollack, "$In_{0.53}Ga_{0.47}As$ Photodiodes With Dark Current Limited by Generation-Recombination and Tunneling," Appl. Phys. Lett., *37* (August 1980), pp. 322–5.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Transmitting Data on the Phase of Speech Signals

By W. C. WONG, R. STEELE, and C. S. XYDEAS

(Manuscript received March 12, 1982)

A method for embedding data into speech signals without recourse to bandwidth expansion is proposed. Sampled speech is assembled into contiguous blocks of N samples and the Discrete Fourier Transform (DFT) is performed on each block. All the phase components in the message band, or the last J components in this band, are discarded when unvoiced or voiced speech is present, respectively. The data is introduced in the place of these rejected phase components, being $+\pi/2$ for a logical 0 and $-\pi/2$ for a logical 1. The magnitude of the coefficients associated with the data-carrying phase components are scaled to guard against data errors resulting from channel noise. The inverse DFT yields the transmitted sequence. The receiver performs the inverse process, stripping off the data and replacing it with random phase values. For an average transmission rate of approximately 1 kb/s and a channel signal-to-noise ratio of 30 dB, the bit error rate was 5.5×10^{-4}, and the average signal-to-noise ratios for voiced and unvoiced speech were 24 and −3 dB, respectively. However, the unvoiced sounds were perceived with negligible distortion owing to the preservation of their magnitude spectra. Modest error-correction codes can be used to reduce the bit error rate to 10^{-7} while maintaining the same recovered speech quality, provided the average transmitted bit rate is decreased to $\simeq$500 b/s.

I. INTRODUCTION

Embedding data in speech signals without a significant enlargement of signal bandwidth has a great attraction if the data can be recovered without error, and the degradation of the speech is perceptually acceptable. There is a euphoria of getting a bargain, almost something for nothing. Of course it is not serendipity, but rather an exploitation of the innate redundancy in speech.

A recent proposal by Steele and Vitello[1,2] for the simultaneous transmission of speech and data signals attempted to preserve the speech signal while accepting a bandwidth expansion of the transmitted signal. In their system the speech conveys the data using the principles of analog speech scrambling. The data becomes the scrambling key, while the receiver acts the part of a code breaker. Every time the code is deciphered correctly the receiver recovers both the data and the speech. Codes are therefore selected that are easy to break. Frequency inversion scrambling was used to achieve data rates of 700 b/s over ideal channels, and 125 b/s when additive channel noise was as high as 10 dB below the mean square value of the speech signal. In both cases there were no data errors associated with the 39,000 speech samples used in the experiments.

We now propose a system for the simultaneous transmission of speech and data that avoids a bandwidth expansion of the transmitted signal compared to that of the original speech signal, but does engender a modest reduction in the perceptual quality of the received speech.

II. THE SYSTEM

The combined transmission of data and speech in our proposed system is achieved by discarding some phase components in the speech signal and replacing them with data. At the receiver the data are removed and replaced with random phase components. By judicious choice of which phase components are used for the conveyance of data, we are able to ensure that the recovered speech quality is only marginally degraded by the presence of the data.

The speech signal bandlimited between 200 Hz and 3.2 KHz is sampled at 8 KHz and divided into sequential blocks each containing N samples. To decide whether a block of samples is to convey data, and if so, how many bits, we perform what is tantamount to a crude voice, unvoiced, or silence detection. The mean square value σ_x^2 of the samples in the block is computed and compared with two thresholds, T_1 and T_2. These thresholds float compared with the mean square value $\sum_x^2$ of the speech calculated over many blocks, such that T_1 and T_2 are α_1 and α_2 dB below $\sum_x^2$, respectively. From inspection of five sentences we experimentally determined that $\alpha_1 = 18.5$, and $\alpha_2 = 30$. The mean square value σ_x^2 is compared with these thresholds and the decision to transmit B_1 or B_2 bits of data is made according to

$$\sigma_x^2 < T_2; \text{ NO DATA TRANSMITTED} \tag{1}$$

$$T_2 \le \sigma_x^2 < T_1; \text{ B}_1 \text{ BITS TRANSMITTED} \tag{2}$$

$$T_1 \le \sigma_x^2; \text{ B}_2 \text{ BITS TRANSMITTED,} \tag{3}$$

where

$$B_2 < B_1. \tag{4}$$

Although our decision as to whether to transmit data, and if so, whether B_1 or B_2 bits will be embedded in the speech signal, depends only on Inequalities (1) to (3), we may consider that to a good approximation these Inequalities refer to the presence of silence, unvoiced speech, or voiced speech, respectively. Observe that as a consequence of Inequalities (1) to (3) the bit rate is variable, being dependent on the presence and nature of the speech signal. As the system is conceived for embedding data into speech signals, we envisage conventional modem techniques being deployed for the transmission of data during prolonged silences,[3,4] assuming a time assignment speech interpolation (TASI)-type arrangement is not in service.

Provided Inequality (2) or (3) is satisfied, the discrete Fourier Transform (DFT) is performed on the block of speech samples $\{x(n)\}_{n=0}^{N-1}$, namely,

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk}; \quad k = 0, 1, \cdots, N-1 \tag{5}$$

or

$$X(k) = Re(k) + jIm(k), \tag{6}$$

where $Re(k)$ and $Im(k)$ are the real and imaginary components of $X(k)$, respectively. The magnitude of $X(k)$ is

$$|X(k)| = \sqrt{Re^2(k) + Im^2(k)} \tag{7}$$

and its phase angle is

$$\phi(k) = \tan^{-1}\left(\frac{\mathrm{Im(k)}}{\mathrm{Re(k)}}\right). \tag{8}$$

The procedure for assigning data depends on whether Inequality (2) or (3) occurs.

2.1 Unvoiced speech

If Inequality (2) is satisfied, the speech is almost certainly unvoiced. When unvoiced speech occurs the vocal cords do not vibrate, and the sounds originate because of turbulent air flow at a constriction in the vocal tract. Unvoiced sound has a noise-like nature and tends to have low energy. The former characteristic is valuable when data, transmitted as the phase components in the unvoiced speech signal, are removed at the receiver and replaced with random phase components. The re-introduced phase components have a similar randomness to the original components, and the perceptual quality of the sound is negligibly degraded. The low energy of unvoiced speech is, by contrast, an undesirable feature when data is embedded in the phase components, as channel noise may precipitate large variations in the phase of the received signal causing a high bit error rate (BER). Consequently, steps must be taken to increase the energy of the unvoiced sounds.

2.1.1 μ-law spectral scaling

The effect of channel noise can be mitigated by scaling the magnitude of the spectral components according to the μ-law,[5] producing magnitude components

$$|D(k)| = \frac{V \log\left\{1 + \mu_{uv}\frac{|X(k)|}{V}\right\}}{\log(1+\mu_{uv})}; \quad |X(k)| < V$$
$$= V; \qquad |X(k)| \geq V, \tag{9}$$

where μ_{uv} is the compression factor for unvoiced speech and V is the μ-law overload parameter. The factor μ_{uv} is selected to provide an acceptably low BER, and also to contain the amplitude range of the transmitted signal. The experimental determination of μ_{uv} is discussed in Section IV. The components $|D(k)|$ are calculated for k spanning the voice bandwidth, i.e., k_{c1} to k_{c2}, where k_{c1} and k_{c2} are the spectral component associated with 200 and 3200 Hz, respectively.

2.1.2 Data insertion

Having described the scaling of the magnitude of the frequency components, we now consider how the data of B_1 bits are embedded in the phase spectrum. All the unvoiced phase components over the speech bandwidth are discarded and replaced with binary phase components determined by the data. As the phase angle $\phi(k)$ is confined to $\pm\ \pi$ radians, we arrange for phase components carrying data to be designated $\theta(k)$ and have values

$$\theta(k) = \pi/2, \text{ signifying logical 0}$$
$$= -\pi/2, \text{ signifying logical 1} \tag{10}$$

for $k = k_{c1}$ to k_{c2}. Although multi-level $\theta(k)$ does increase the amount of data embedded in the speech blocks, we opted for binary $\theta(k)$ to make the system more robust to channel impairments. Unless otherwise stated we will assume that every phase components contains a data bit, whence

$$B_1 = k_{c2} - k_{c1} + 1. \tag{11}$$

However, in the presence of channel impairments we may allocate each bit to an odd number of phase components, and decide on the logical value of the bit at the receiver by a simple majority vote of the logical values associated with the received phase components. More complex channel coding techniques can be employed to further reduce BER.

2.1.3 The combined data and unvoiced speech sequence

The combined speech and data signal is obtained by performing the inverse discrete Fourier transform (IDFT) on the magnitude and phase spectral components. The original spectral coefficients are

$$X(k) = |X(k)|e^{j\phi(k)} \tag{12}$$

and those that carry data are

$$D(k) = |D(k)|e^{j\theta(k)}, \tag{13}$$

where $|D(k)|$ is given by eq. (9). The combined data and speech sequence is

$$\begin{aligned} g(i) = \frac{1}{N}\Bigg[& \sum_{k=0}^{k_{c1}-1} X(k)e^{j\frac{2\pi}{N}ik} + \sum_{k=k_{c1}}^{k_{c2}} D(k)e^{j\frac{2\pi}{N}ik} \\ & + \sum_{k=k_{c2}+1}^{N-2-k_{c2}} X(k)e^{j\frac{2\pi}{N}ik} + \sum_{k=N-1-k_{c2}}^{N-1-k_{c1}} D(k)e^{j\frac{2\pi}{N}ik} \\ & + \sum_{k=N\text{-}k_{c1}}^{N-1} X(k)e^{j\frac{2\pi}{N}ik}\Bigg]. \end{aligned} \tag{14}$$

Some values of $\theta(k)$ are used to inform the receiver whether Inequality (2) or (3) applies. This side information only constitutes a minor part of the transmitted data. The receiver is able to determine if Inequality (1) is valid by examining the mean square value of the received signal blocks.

2.2 Voiced speech

In voiced speech the vibration of the vocal cords causes broad spectrum puffs of air to excite the vocal tract, and the short-time Fourier spectrum of the speech has a quasi-periodicity, and an energy level considerably in excess of that encountered with unvoiced speech. Consequently, if data is loaded onto too many phase components, the quasi-periodicity of the recovered voiced speech will be disturbed and the speech quality degraded. We therefore discard only J phase components,

$$J < k_{c2} - k_{c1}, \tag{15}$$

whenever Inequality (3) is satisfied, and replace them with B_2 bits. Typically, J is 0.16 to 0.33 of $k_{c2} - k_{c1}$. Of course, Inequality (3) may sometimes occur when unvoiced speech is present, but only J phase components will be used for the conveyance of data. Although the occurrence of Inequality (3) signifies that fewer phase components are available for data transmission, this state is important as voiced speech is approximately four times more prevalent than unvoiced speech. The maximum value of B_2 is J, and this will be assumed to occur unless otherwise stated. However, we note that if we assign error protection coding to the binary phase components carrying data, B_2 decreases, and so does the BER compared to the situation when B_2 is equal to J.

Spectral scaling using μ-law is also necessary for voiced speech to ensure that for the expected channel noise the BER is negligible. Equation (9) is applicable, where $|D(k)|$ is determined over the range of J coefficients. The combined voiced speech and data sequence conforms to eq. (14) with the exception that $D(k)$ extends over the range of J components.

The block diagram for embedding data into the speech signal is displayed in Fig. 1. The speech signal sampled at f_s is directed into either shift register SR_1 or SR_2, with switches S_1 and S_2 changing their positions every N/f_s seconds. While the speech samples are being extracted from SR_1, say, the computation of the mean square value σ_x^2 of the current N speech samples entering SR_2 is in progress. After N/f_s seconds σ_x^2 is determined and compared with parameters T_1 and T_2. If Inequality (1) prevails the output of comparators COMP.1 and COMP.2 are logical 0 and logical 1, respectively, and consequently the NOR gate is in the logical 0 state. Both switches S_3 and S_4 receive logical 0 signals and remain open, preventing data from being placed into the data store. When switches S_1 and S_2 change, the logical 1 state of COMP.2 is also used to inhibit the speech samples from entering the Fast Fourier Transform (FFT) circuit, and instead routes the contents of SR_2 directly to the output, bypassing the data-embedding system. This later arrangement is not shown in Fig. 1. Should Inequality (3) occur, COMP.1, COMP.2, and the NOR gate occupy logical 1, 0, and 0 states, respectively. Switch S_3 closes, and (B_2-ϵ) bits are passed into the data store, where ϵ is employed to inform the receiver that Inequality (3) applies. If both COMP.1 and COMP.2 are in the logical 0 state, the NOR gate becomes a logical 1, closing switch S_4. Data of (B_1-ϵ) bits proceed via switch S_4 into the data store, where this time ϵ signifies the presence of Inequality (2). Thus, ϵ need be only one bit, unless protection coding is added. If speech is deemed to be present, the speech in SR_2 is applied to the FFT device, and the magnitude $|X(k)|$ and phase $\phi(k)$ components of the block of speech samples generated. The $|X(k)|$ and $\phi(k)$ components are passed via switches S_5 and S_7 either directly to the Inverse Fast Fourier Transform (IFFT) via switches S_6 and S_8, or are subjected to spectral scaling and data insertion according to the number B_2 or B_1 bits removed from the data store. If voiced speech occurs only J components of $\phi(k)$ are

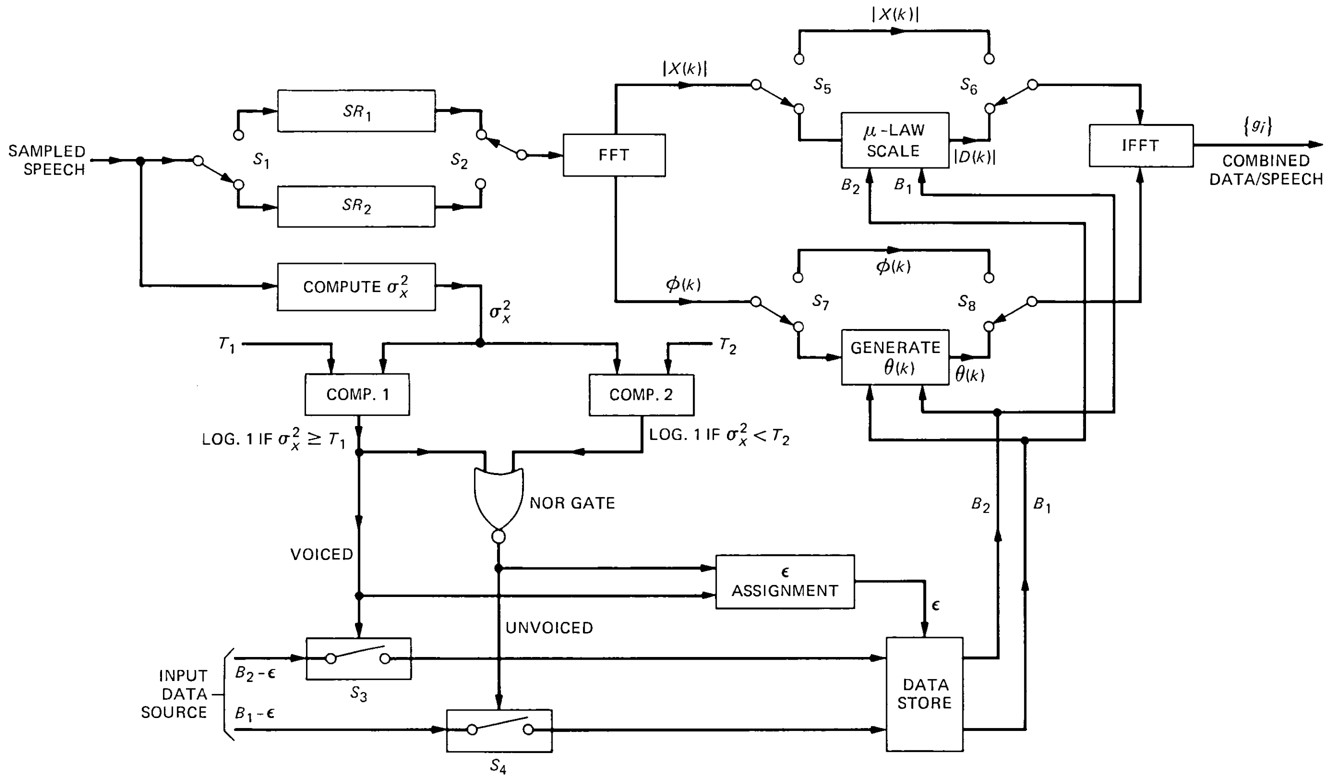


Fig. 1—Block diagram for embedding data into the phase components of the speech signal.

used, but when B_1 is present $(k_{c2} - k_{c1})$ components of $\phi(k)$ are converted to $\theta(k)$. Observe that the spectral component(s) for ϵ is always located in the same location in the J spectral region. The sequences at the output of switches S_6 and S_8 are applied to the IFFT to yield the combined speech and data sequence $\{g_i\}$.

III. THE RECEIVER

We will refrain from discussing the numerous methods by which the combined speech and data signal can be transmitted, nor will we address the variety of channels, their attendant equalization, or the techniques of correctly locking the receiver clock and the attendant acquisition of sample and block synchronization. Rather, we will assume that the combined signal is correctly sampled and ordered into the correct blocks.

The receiver's first task is to remove the data, but before that the receiver must ascertain if data have been transmitted. This is relatively straightforward since if data are embedded in the speech block, spectral scaling of the coefficients will have been performed at the transmitter, and the mean square value of the combined signal is significantly greater than T_2 of Inequality (1). If no data is considered to be present, the received signal is accepted as the received speech signal. When data is deemed to be present, we need to determine whether it is located in J or $k_{c2} - k_{c1}$ phase components. Accordingly, the FFT is taken of the combined data and speech sequence, and the spectral phase component(s) associated with the ϵ bit(s) examined so the receiver can determine if Inequality (2) or (3) applies. Once this is accomplished the data are extracted from the received phase components $\hat{\theta}(k)$ that are known to contain binary information, according to

$$0 \le \hat{\theta}(k) < \pi, \qquad \text{logical 0 generated}$$

and

$$-\pi \le \hat{\theta}(k) < 0, \qquad \text{logical 1 generated.} \tag{16}$$

Having removed the data we proceed to recover the speech signal. The missing phase components are replaced by phase components having any value between $\pm\pi$ with equal probability. Those coefficients whose magnitudes were scaled are then de-scaled by inverse μ-law operation. The IFFT follows, and the speech sequence $\{\hat{x}_i\}$ so formed contains distortion, which is most serious near the ends of the blocks. A simplified explanation of this distortion is as follows. Consider successive DFT spectra to contain one line, with the phase of this line changing every block by $\pi/2$, while the amplitude of the spectral lines remains constant. The time waveform is a sinusoid whose phase changes by $\pi/2$ at the ends of the blocks. Now consider many spectral components whose phase changes by a random value between blocks. We may again consider these components to be transformed into the time domain as sinusoids with abrupt phase changes at the block boundaries. In the case of speech, each spectral component has a different magnitude, and J or all the components may have their phases randomized. The end of block distortion ensues, its values varying from one block boundary to another in a manner difficult to quantify.

To mitigate end of block distortion we apply median filtering[6,7] to those samples at the ends of adjacent blocks. Thus, samples between the mth and $(m + 1)$th blocks are median filtered to give

$$\tilde{x}_{mN+j+i} = \underset{j=-M}{\overset{M-(2i-1)}{\mathrm{MED}}} \{\hat{x}_{mN+j}, \hat{x}_{mN+j+1}, \cdots, \hat{x}_{mN+j+i}, \cdots, \hat{x}_{mN+j+2i}\}, \tag{17}$$

where i is a constant for a particular filter whose length is

$$L = 2i + 1, \qquad i = 1, 2, 3, \cdots, \tag{18}$$

i.e., the median value of L samples is the filtered sample. The number of samples median filtered in the vicinity of the block boundaries is

$$\lambda = 2(M + 1 - i) \tag{19}$$

and the number of samples used in the filtering of λ samples is

$$\gamma = 2(M + 1), \tag{20}$$

where M is a system parameter.

As an illustration of how the median filter equations are used, consider the example of a three-point median filter, $L = 3$, and $M = 5$. Equation (17) becomes for these parameters

$$\tilde{x}_{mN+j+1} = \underset{j=-5}{\overset{4}{\mathrm{MED}}} \{\hat{x}_{mN+j}, \hat{x}_{mN+j+1}, \hat{x}_{mN+j+2}\}, \tag{21}$$

and $\tilde{x}_{mN+j+1}$ is the median value of $\hat{x}_{mN+j}$, $\hat{x}_{mN+j+1}$ and $\hat{x}_{mN+j+2}$. The number of samples used in the filtering process is $\gamma = 12$, which will be made up of six samples at the end of the mth block and six samples at the commencement of the $(m + 1)$th block. There are $\lambda = 10$ samples median filtered commencing with $\tilde{x}_{mN-4}$ when $j = -5$, and terminating with $\tilde{x}_{mN+5}$ when $j = 4$. Thus, the number of terms L in the brackets of Eq. (17) gives the number of samples used in the filtering process of each sample. As j steps from $-M$ to $M - (2i - 1)$, the sample being filtered, namely $\hat{x}_{mN+j+i}$, also changes under the control of j.

After λ samples have been filtered, the recovered speech sequence is obtained, having these λ samples, and $N - \lambda$ components from $\{\hat{x}_i\}$ for each block of N samples. The median filtering significantly reduces the end of block distortion.

IV. RESULTS

The sentences, "Glue the sheet to the dark blue background," "Rice is often served in round bowls," "Four hours of steady work faced us," and "The box was thrown beside the parked truck," were used in our experiments. The first two sentences were spoken by females, the remainder by males. These concatenated sentences constituting our speech signal were bandlimited between 200 Hz and 3200 Hz and sampled at 8 KHz to provide the input speech sequence, $\{x_i\}$. Random binary data were introduced into the phase components of $\{x_i\}$ in the manner described in Section II. The information ϵ was assumed to be received without error. This is a reasonable assumption because ϵ can be specified by one bit and as the error rate of the phase components carrying data will be shown to be 0.055 percent, the probability of ϵ being in error can be rendered negligible by assigning a small number of error-correcting bits to ϵ. The block size N was 256.

Because of the spectral scaling, the first experiments related to the increases in the peak and rms values of the combined speech and data sequence, $\{g_i\}$, compared to those of the input speech sequence, $\{x_i\}$. We were concerned that the spectral scaling might significantly increase the amplitude and power levels of the original speech signal and overload the communication channel. To observe the effect of spectral scaling on the amplitude components in $\{g_i\}$ we proceeded as follows. Each block of speech was examined, and those blocks where Inequality (3) applied, our so-called voiced blocks, were noted. Using these blocks we calculated two signal expansion parameters, which we defined as,

$$r_v \triangleq \frac{1}{\psi_v} \sum_{i=1}^{\psi_v} \frac{|g|_{\max,v,i}}{|x|_{\max,v,i}} \tag{22}$$

and

$$\rho_v \triangleq \frac{1}{\psi_v} \sum_{i=1}^{\psi_v} \frac{\Omega_{g,v,i}}{\Omega_{x,v,i}}, \tag{23}$$

where $|g|_{\max,v,i}$ and $|x|_{\max,v,i}$, and $\Omega_{g,v,i}$ and $\Omega_{x,v,i}$ are the maximum and rms values of the combined speech and data sequence, and the input speech sequence, in the ith blocks, respectively. The number of voiced blocks is ψ_v, where the subscript v is used to signify the applicability of Inequality (3). Figures 2 and 3 show the variation of r_v and ρ_v as a function of the spectral scaling factor, μ_v, for voiced speech. As more phase components are used to convey data, i.e., increasing J, more spectral magnitude components are increased by μ-law spectral scaling; and on performing the IFFT the rms and maximum amplitudes of the voiced blocks in the transmitted signal are increased, and hence r_v and ρ_v are increased for a given μ_v. Similarly, for a given J the effect of

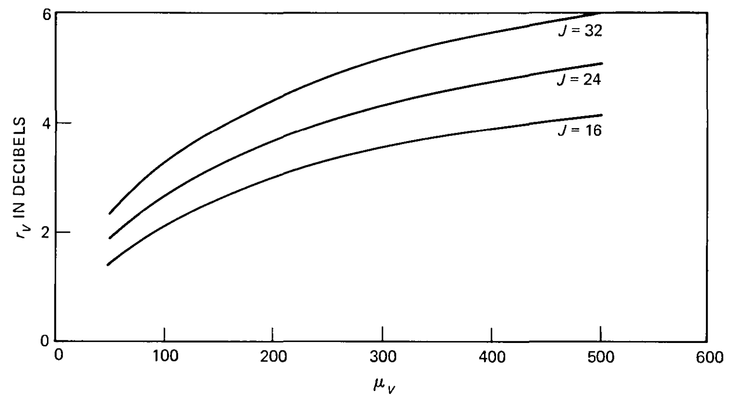


Fig. 2—Variation of the average ratio of maximum amplitudes of the transmitted signal to the input speech signal, r_v, as a function of the μ-law scaling factor, u_v, for voiced speech for different values of J.

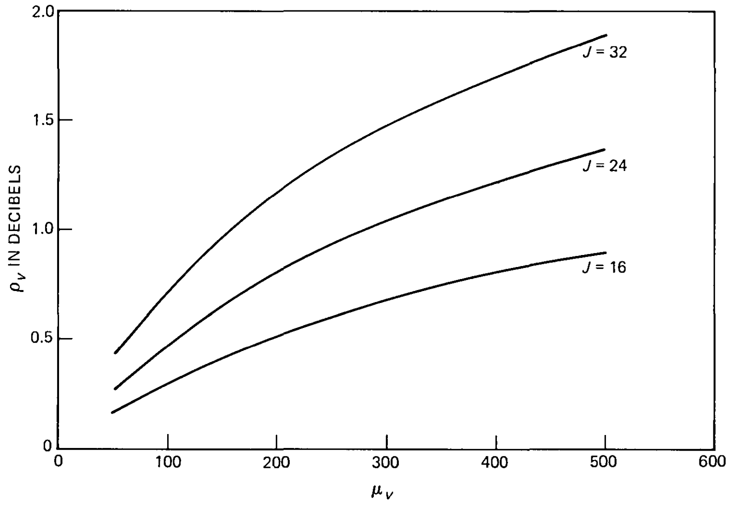


Fig. 3—Variation of the average ratio of the rms value of the transmitted signal to the input speech signal, ρ_v, as a function of the μ-law scaling factor u_v for voiced speech for different values of J.

increasing μ_v is to increase the spectral scaling of the magnitude components, which consequently increases r_v and ρ_v.

By selecting only those blocks where Inequality (2) applied, we found the signal expansion factors r_{uv} and ρ_{uv} using the same procedures as employed for r_v and ρ_v [see eqs. (22) and (23)]. The subscript *uv*, an abbreviation for unvoiced speech, implies the validity of Inequality (2). The variation of r_{uv} and ρ_{uv} as a function of the spectral scaling factor, μ_{uv}, for unvoiced speech is displayed in Fig. 4. Unlike

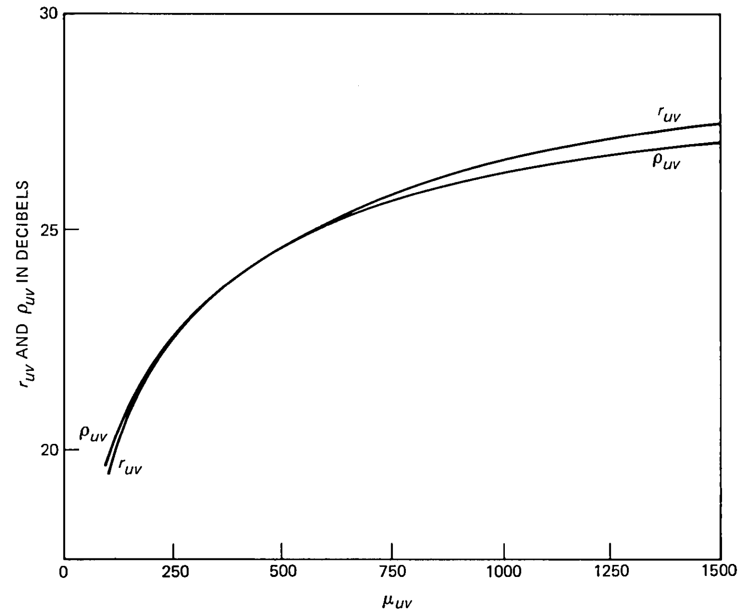


Fig. 4—Variation of r_{uv} and ρ_{uv} as a function of μ_{uv}.

the small increases in r_v and ρ_v that occur for voiced speech, the effect of spectral scaling all the magnitude components by large values of μ_{uv} results in substantial increases in r_{uv} and ρ_{uv}. However, unvoiced speech has much lower magnitude and rms values than voiced speech, enabling much larger values of r_{uv} and ρ_{uv} to be used, thereby protecting the data from channel noise. If it is required that the peak or rms value of $\{g_i\}$ is not to exceed that of $\{x_i\}$, an attenuator must be placed after the IFFT in Fig. 1. The effect of such an attenuator on the recovered signal-to-noise ratio (s/n) and BER will be discussed later.

An objective criterion for the quality of the recovered speech signal should take cognizance of the particular process being used to convey data, yet be sufficiently well known to have comparative value. In this system the distortion in the recovered speech signal originates from two main processes, namely, the randomizing of the phase components of voiced and unvoiced speech, and the effect of additive channel noise. The randomization of the phase components does not alter the recovered magnitude spectra, and thus spectral distortion measures[8] based on spectral power are inappropriate. The only errors in the magnitude spectra derive from the channel noise. Signal-to-noise ratio measurements are familiar to engineers in spite of their shortcomings, and the two most widely quoted are the average s/n and the segmental s/n.[9] In the former the ratio of the average signal power to the average error power is found. In determining segmental s/n the signal is divided into segments or blocks, and the average signal to average error power is computed in decibels for each segment. Then s/n values of each segment are averaged to give segmental s/n. We decided to use segmental s/n, and proceeded to divide the speech into voiced or unvoiced segments. We made this division because the effect of randomizing the phase yields s/n values for a segment that critically depend on whether the segment contains voiced or unvoiced speech. A low s/n for unvoiced speech can be anticipated, as randomizing every phase component yields a time waveform that is radically different from the original segment of speech. The s/n for that block of speech is accordingly very low, and is often negative. However, because the magnitude of the spectra for the recovered and original speech signals are the same, these signals are perceived to be similar. In the case of voiced speech the effect of randomizing J spectral components without altering their magnitudes results in end of block distortion. This distortion is mitigated by employing median filtering as previously described. The end of block distortion is not significant with unvoiced speech because of the relatively small magnitudes of the spectral coefficients. By measuring the s/n of each voiced segment, we provide a measure of the end of block distortion. Thus, segmental s/n is a reasonable measure for voiced speech, and a poor measure for unvoiced speech, in that the value of the segmental s/n has a close correspondence with the perceived speech in the case of voiced speech, and vice versa for unvoiced speech. We note in passing that in waveform encoding, like the situation here, the segmental s/n is usually high for voiced speech and low or negative for unvoiced speech.[10]

In our experiments we proceeded as follows. Assuming the channel to be ideal we determined the s/n of the recovered speech signal as a function of the number J of phase components discarded for voiced speech. Only blocks where Inequality (3) applied were used in the s/n calculation. We performed experiments for J measured over different coefficient ranges, e.g., from k_{c1} to higher values of k, about the center of the coefficient range, and from k_{c2} to lower values of k. The location of the range of J caused different perceptual impairments in the recovered speech signal. From informal listening tests we concluded that the latter range for J was preferable, and, accordingly, we display in Fig. 5 the s/n for voiced speech as a function of J measured from k_{c2} to lower values of k. In determining the s/n, we employed the median filter having a length L of 3, and $M = 5$. As the curve in Fig. 5 was obtained for an ideal channel, it is independent of the values of μ_v and μ_{uv}, factors introduced to avoid data errors in the presence of channel impairments. The exchange of s/n in decibels with J is given by

$$\text{s/n} \simeq 39 - 0.375J; \qquad 8 \leq J \leq 40, \tag{24}$$

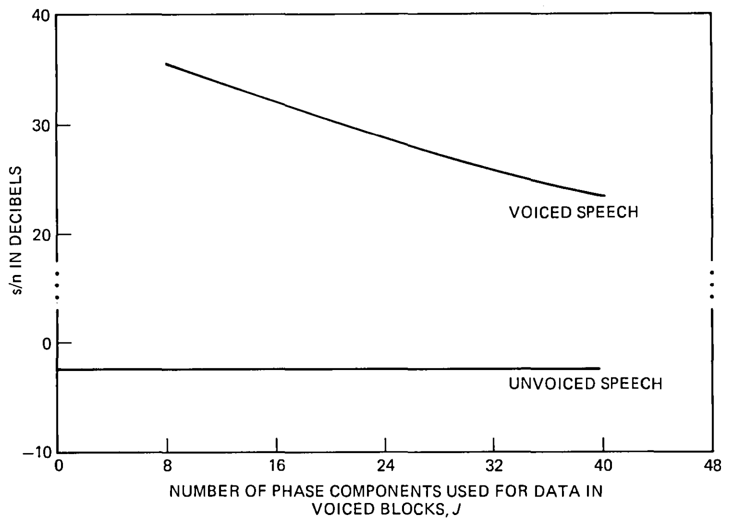


Fig. 5—Variation of s/n versus J for voiced speech. The s/n of unvoiced speech is also shown.

i.e., a loss of 0.375 dB in s/n per phase component of voiced speech employed for the transmission of data.

In the case of blocks containing unvoiced speech 98 phase components in each block are used for the conveyance of data (corresponding to 200 to 3200 Hz, $N = 256$), and the segmental s/n for these blocks is only −2 dB. As mentioned above, this low s/n for unvoiced speech was expected owing to the randomization of all the phase components in the recovered speech. However, the perceptual quality and intelligibility of the recovered unvoiced speech is good as the magnitude spectra are maintained, and the excitation in unvoiced speech is noise-like. The s/n for unvoiced speech is shown in Fig. 5 as a horizontal line.

For the sentences used, transmitted data rates of 1200, 1024, and 852 b/s were achieved for $J = 32$, 24, and 16, respectively.

To prevent the channel from being overloaded by excessive amplitude levels resulting from signal amplification owing to spectral component scaling, attenuation of $\{g_i\}$ was performed. The attenuation was adjusted until the range of amplitude levels of the combined speech and data signal was the same as that of the input speech signal. Specifically, we found the block with the largest output amplitude whose magnitude expansion parameter was r_k, say. The attenuation in decibels was then set at

$$I = 20 \log_{10}(r_k)$$

for the whole speech signal. Channel noise $\{n_i\}$ was next added to the transmitted signal, and for a constant channel noise power of minus P

dB below the mean square value of the input speech signal, the change in s/n relative to the s/n in the absence of spectral scaling was found as a function of μ_v for blocks where Inequality (3) applied. This was repeated for different values of P and two values of J to yield the curves shown in Fig. 6a and b. As we expected, when P becomes progressively more negative, the change in s/n, namely Δs/n, approaches zero for all μ_v. When the additive noise power P is high and $J = 32$, there is a loss in s/n that increases with μ_v, but never exceeds 3 dB for the parameters shown in Fig. 6a. For $J = 24$, the loss in s/n is much smaller ($\simeq$1 dB), and for μ_v below 100, Δs/n may be slightly positive. This small positive value of Δs/n arises because for low values of μ_v, the spectral scaling of the J coefficients carrying data is insufficient to cause the combined data and speech sequence $\{g_i\}$ to be attenuated. Thus, for a given channel noise power P the channel s/n decreases with the result that the recovered s/n is marginally enhanced. When μ_v exceeds 100, and the attenuation of $\{g_i\}$ is as described above, the channel s/n decreases, and Δs/n takes on negative values. When the experiment was repeated with blocks where Inequality (2) applied, Δs/n was always positive as shown in Fig. 6c. Observe that for unvoiced speech no attenuation of the combined speech and data signal need be imposed for $\mu_{uv} \leq 400$, as the signal does not exceed the levels found in voiced speech. Consequently, Δs/n is nearly constant until $\mu_{uv} > 400$, whence attenuation of $\{g_k\}$ is employed. Δs/n decreases slowly with μ_{uv}, and Δs/n is marginally greater for P of −20 dB than −30 dB, i.e., −20 dB of channel noise is advantageous. However, the variation of Δs/n in Fig. 6 is not great, being positive for unvoiced speech and negative (in general) for voiced speech.

With the attenuator adjusted as previously described such that the amplitude range of the transmitted signal and the original speech signal are the same, we observed an improvement in BER, defined as

$$IBER = 20 \log_{10}\left\{\frac{BER_\mu}{BER_o}\right\}, \tag{25}$$

where BER_μ and BER_o represent the BER when spectral scaling of value μ is used and when no spectral scaling is employed, respectively. The variation of IBER with μ for different values of noise power P is displayed in Fig. 7. The smallest value of P employed was −30 dB, as we did not have sufficient data for reliable results when P was more negative. We see from Figs. 7a and b that μ_v of the order of 250 is a good choice as it provides a large value of IBER while avoiding significant losses in s/n, as displayed in Figs. 6a and b. Thus, for $J = 24$, $P = -30$ dB, a $\mu_v = 250$ provides a gain in BER of $\simeq$50 dB while sustaining a loss in recovered speech s/n of only 0.5 dB. As is expected,

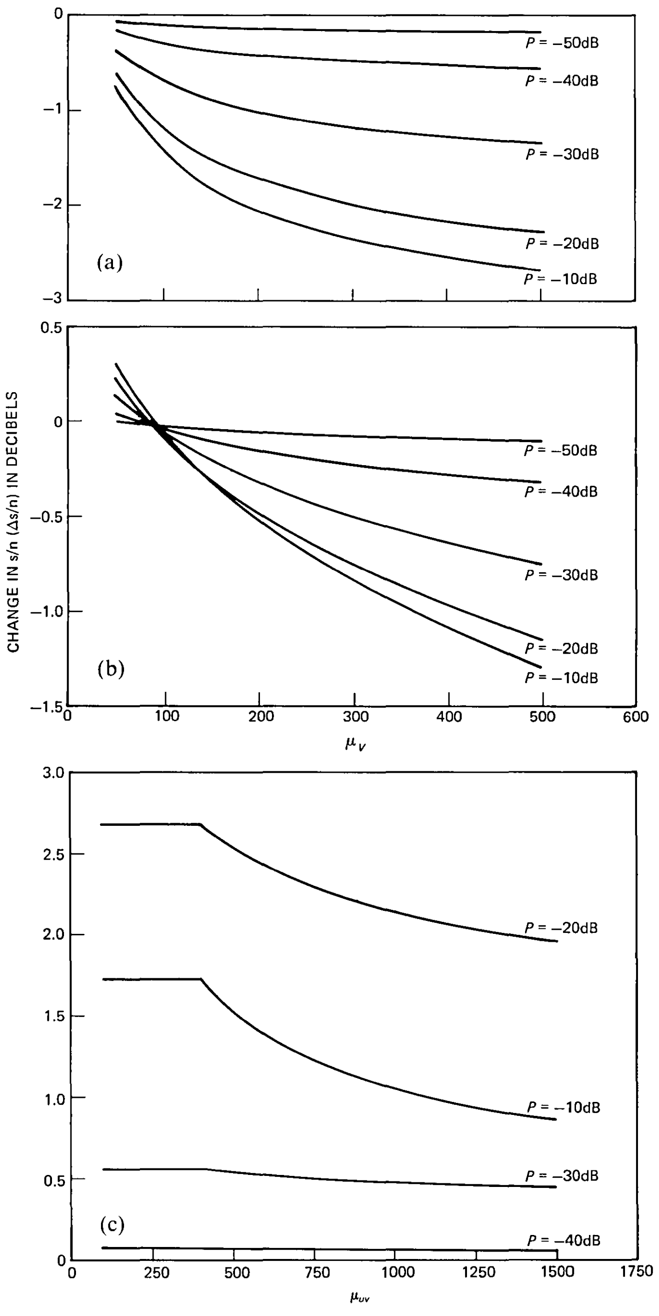


Fig. 6—Effect of spectral scaling on s/n. The change (Δs/n) in s/n relative to the s/n in the absence of spectral scaling, for different channel noise power P. (a) Δs/n versus μ_v, $J = 32$. (b) Δs/n versus μ_v; $J = 24$. (c) Δs/n versus μ_{uv}.

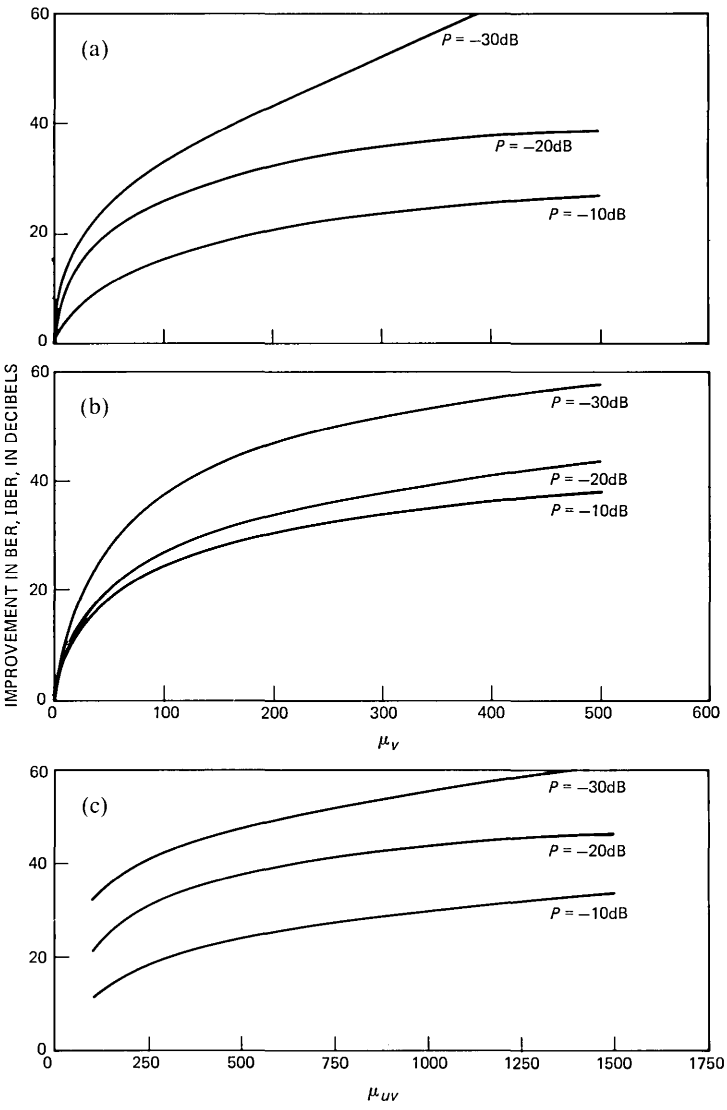


Fig. 7—Improvement in BER, namely IBER, due to spectral scaling, for different channel noise power P. (a) IBER versus μ_v, $J = 32$. (b) IBER versus μ_v, $J = 24$. (c) IBER versus μ_{uv}.

larger values of μ_{uv} apply as shown in Fig. 7c, and a good choice of μ_{uv} is 750. Using this μ_{uv} value, $P = -30$ dB, we achieved IBER of $\simeq$50 dB and a gain in s/n of 0.5 dB. Figures 6 and 7 highlight the desirable properties of spectral scaling, a large improvement in BER, and at worst a small loss in speech s/n.

The channel s/n was computed as

$$s/n_c = 10 \log_{10} \left\{ \frac{\sum_{i=1}^{W} \tilde{g}_i^2}{\sum_{i=1}^{W} n_i^2} \right\}, \tag{26}$$

where $\tilde{g}$ is the attenuated version of g, and W is the number of speech samples in the input speech signal. Although the same noise source was used as in the previous experiments, and the attenuator was employed, s/n$_c$ differs from the s/n of P dB computed using the input speech and the noise signal. This difference arises because $\{\tilde{g}_i\}$ is not identical to $\{x_i\}$. The sequence $\{\tilde{g}_i\}$ depends on μ_v and μ_{uv}, parameters which affect both r and ρ [see eqs. (22) and (23)]. However, the s/n differences are small, and typically are <3 dB. Figure 8 displays the

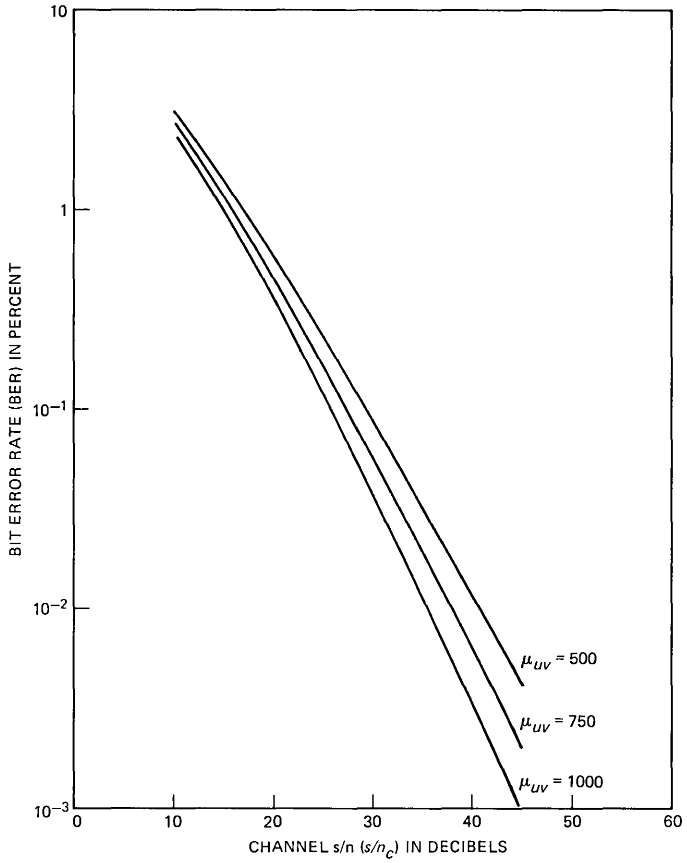


Fig. 8—Variation of BER as a function of s/n$_c$ for different values of μ_{uv}; μ_v = 250, J = 24.

variation of BER as a function of s/n$_c$ for different μ_{uv}, and $\mu_v = 250$. As we anticipated from Fig. 7c, increasing μ_{uv} from 500 to 1000 results in an increase in IBER, which means a decrease in BER. Further increases in μ_{uv} will decrease BER, but the reduction will not be great.

We observe from Fig. 8 that for s/n$_c$ of 30 dB, the BER is 0.055 percent, and as previously stated, the average transmitted bit rate for $J = 24$ is 1024 b/s. This BER can be reduced by using error-correcting codes. For example, if a BCH code is employed such that the number of error-correcting code bits equals the number of data bits, i.e., the average transmission rate is 512 b/s, the BER decreases to approximately 10^{-5} percent. The extent of the trading of the reduction in the average transmitted bit rate for improvements in BER depends on system requirements. In digital radio transmission outage occurs when the BER exceeds 0.01 percent.

The variation of the segmental s/n of the recovered speech signal against s/n$_c$ is shown in Fig. 9 for three values of J, and $\mu_v = 250$ and $\mu_{uv} = 750$. Only blocks satisfying Inequalities (2) and (3) were used in this calculation of segmental s/n. As s/n$_c$ approaches 50 dB we approximate to the ideal channel condition, and by comparing the s/n values with those in Fig. 5 we may observe the deleterious effect of the unvoiced speech s/n on the overall s/n. Thus, for $J = 16$, 24, and 32, the s/n = 25, 23, and 20.5 dB in Fig. 9, whereas when only voiced speech is present the corresponding s/n = 32, 28, and 26 dB. However, the perceptual quality of the recovered speech is more suitably represented by the segmental s/n for voiced speech than by the combined segmental s/n. Thus, the s/n values in Fig. 9 are lower than would be anticipated for the quality obtained. If no phase components had been used for the transmission of data, the variation of s/n with s/n$_c$ would be a straight line at 45 degrees, shown in Fig. 9. The offset of this line from the origin is due to the s/n of the speech being calculated as segmental s/n,[9] and s/n$_c$ is computed according to eq. (26).

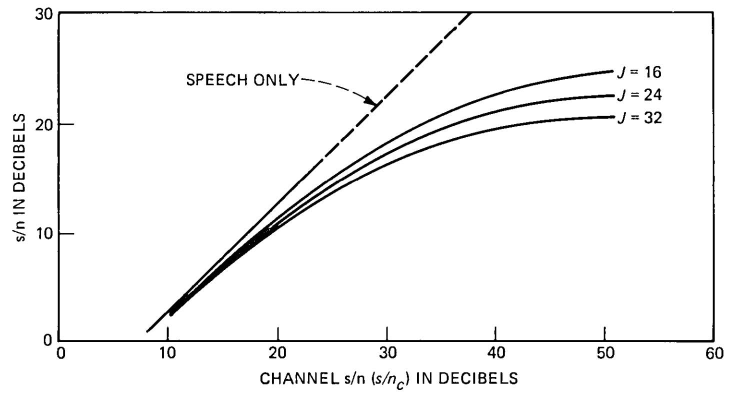


Fig. 9—Recovered s/n as a function of s/n$_c$ for $J = 16$, 24, and 32, $\mu_v = 250$, $\mu_{uv} = 750$.

We conclude this section by providing some waveforms and spectra of the input, transmitted, and recovered signals for $\mu_v = 250$, $\mu_{uv} = 750$, $J = 24$, and $N = 256$. In Fig. 10a twenty blocks of an arbitrary speech signal are shown, having two blocks of intraconversational silence. The combined speech and data sequence $\{g_i\}$ prior to attenuation is displayed in Fig. 10b, where it can be observed that the power level of the unvoiced speech is considerably amplified; high-amplitude, high-frequency components have been introduced into the voiced segments; and those parts of the silence that resided in blocks substantially occupied by voiced speech are carrying data. The recovered speech signal is displayed in Fig. 10c for the case of an ideal channel. The effect of replacing the data-carrying phase components by random ones does not cause serious degradations in the perceptual quality of the recovered speech.

The magnitude of the spectral components of the waveforms in Figs. 10a and b are shown in Figs. 11a and b, respectively. As data is carried by the phase components in the speech signal, the magnitude spectra of the waveforms in Fig. 10a and c are identical. The μ-law scaling of 24 components for voiced speech is seen to substantially enhance its high-frequency components, whereas all 98 components across the speech band are scaled for the unvoiced speech. The μ-law scaling for voiced speech is seen to be reminiscent of frequency pre-emphasis.

V. DISCUSSION

A system has been proposed for the simultaneous transmission of speech and data on the phase of the speech signals, where the bandwidth of the transmitted signal is contained relative to that of the original speech. We knew at the outset that if data was to be conveyed on the phase of speech signals, the receiver would be forced to introduce phase components to replace those that had been discarded at the transmitter in favor of data. We postulated that if the introduced phase components were derived from a random number source, and that their values were confined between $\pm\pi$, then the perceptual degradation in speech quality might be acceptable. Our decision to randomize the values of the introduced phase components at the receiver was based on the knowledge that the variations in the values of the phase spectral components in speech, particularly unvoiced speech, tend to have random behavior. Further, the effect of phase distortion on monaural speech intelligibility is known to be small, the controlling factor being the amplitude spectra. Accordingly, we did experiments, and from informal listening experiences concluded that the randomization of all the phase components of unvoiced speech did not cause serious perceptual degradation. In the case of voiced speech we discovered that if too many phase components were randomized

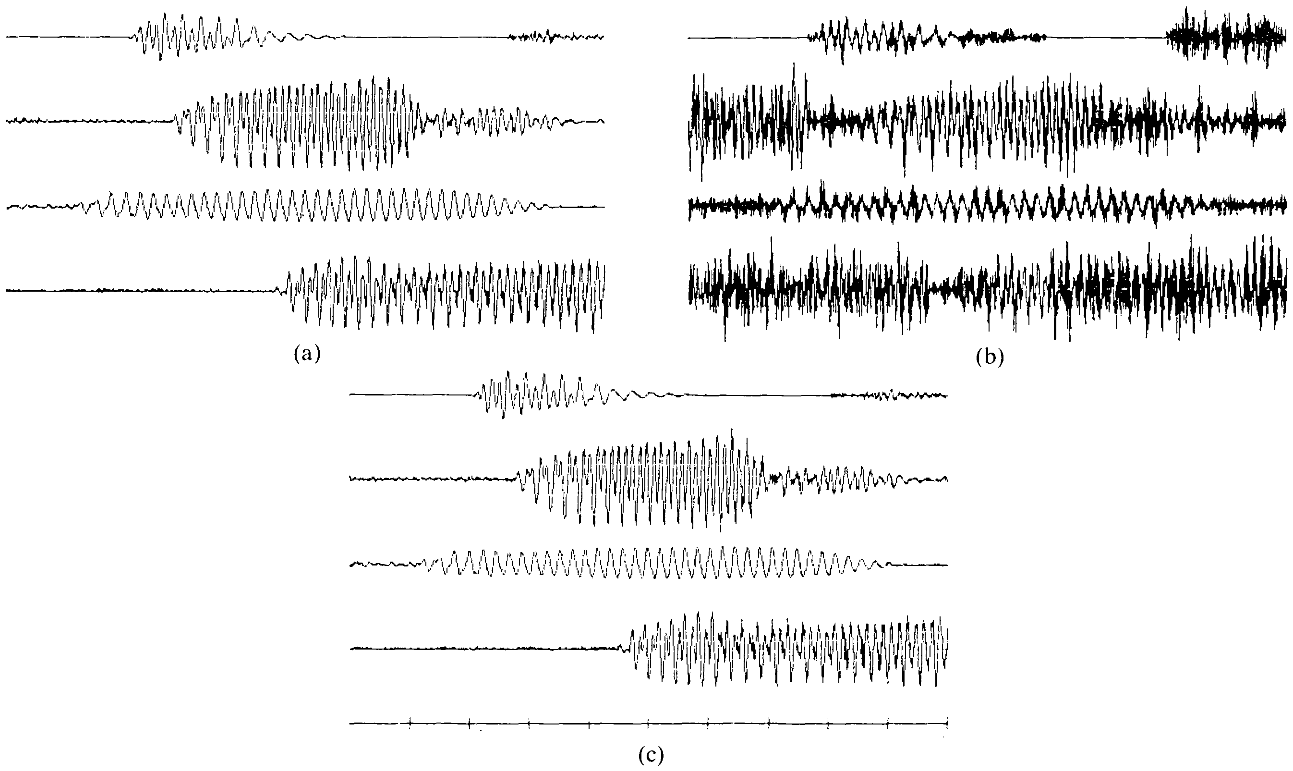
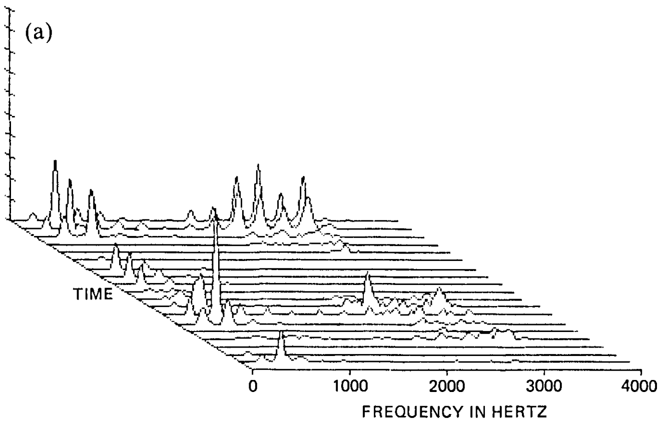


Fig. 10—(a) Arbitrary speech waveform. (b) Combined speech and data signal prior to attenuation. (c) Recovered speech signal; transmission via an ideal channel.



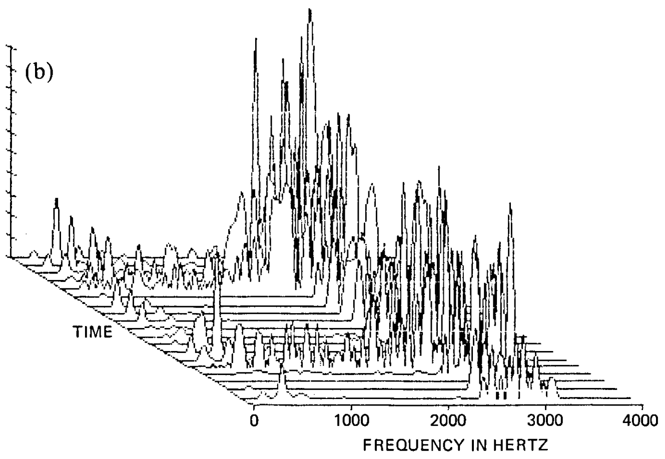


Fig. 11—Magnitude spectra. (a) The signal in Fig. 10a and c. (b) The signal in Fig. 10b.

the recovered speech quality was poor, and out of the 98 phase components available in our experiments we concluded that the maximum number J of phase components that could be randomly perturbed was 32. The position of the J components had different perceptual effects, and we decided to make J span the range of the highest inband frequency components, although the actual position of J is far less important than its value.

Deeming that randomization of the phase components as described was perceptually tolerable, particularly in the presence of channel noise when the channel s/n was approximately 30 dB, we decided to trade the loss in perceptual quality for the implantation of data into those components we had randomized. By this strategy, and for a channel s/n of 30 dB we have been able to achieve an average data rate of 1 Kb/s on the assumption that one bit is assigned to each data-carrying phase component. To achieve this data rate we are required to tolerate a BER of 0.055 percent and an average s/n for voiced and unvoiced speech of 24 and −3 dB, respectively, the measurements being made over four sentences of speech. Observe that the BER can be reduced to a value acceptable for the user by applying channel-coding strategies that result in a reduction in the transmitted bit rate. An example of such a trade-off is given in Section IV. The recovered speech is below toll quality, but the ability to transmit data may make this quality reduction acceptable in certain situations.

Making comparisons of this technique of conveying data on the phase of the speech signal with those employing scrambling methods[1,2] is difficult because of the radically different approaches of these schemes. Embedding data in speech by scrambling can be made to have a very small bandwidth expansion by suitable choice of scrambling code and block size.[2] Increasing the complexity of the scrambling algorithm and the number of bits per block of speech scrambled alters the systems performance in a way that is difficult to predict.

The previously described system using scrambling techniques,[1,2] and the one described here have only been evaluated for noisy channels. Which system would perform best in an actual communications network, and what the requirements would be on channel equalization and synchronization are unknown quantities. What we can say is that errors in the samples at the receiver attributable to noise or imperfect channel equalization, the presence of an unwanted sample, and the loss of a wanted one owing to incorrect synchronization, are smeared over the spectral components by the DFT. The data here is binary and therefore considerable noise on the data-carrying phase can be tolerated. By using error detection and correction coding the data rate can be sacrificed to a value commensurate with a specified BER for a given set of channel impairments.

However, our quest was not to investigate the numerous channel conditions. It was to determine if speech and data could be transmitted over a noisy channel by embedding the data in the phase of speech, and further, if the transmitted bit rate could be sufficiently high to be useful, the BER acceptably low, and the degradation in the recovered speech quality perceptible but not annoying. Our conclusion is affirmative.

REFERENCES

1. R. Steele and D. Vitello, "Simultaneous Transmission of Speech and Data Using Code-Breaking Techniques," B.S.T.J., *60,* No. 9 (November 1981), pp. 2081–2105.
2. R. Steele and D. Vitello, "Embedding Data in Speech Using Scrambling Techniques," ICASSP '82, Paris, *3* (May 1982), pp. 1801–4.
3. P. T. Brady, "A Statistical Analysis of On-Off Patterns in 16 Conversations," B.S.T.J., *47,* No. 1 (January 1968), pp. 73–91.

4. M. R. Schroeder and S. L. Hanauer, "Interpolation of Data with Continuous Speech Signals," B.S.T.J., *46,* No. 8 (October 1967), pp. 1931–3.
5. B. Smith, "Instantaneous Companding of Quantized Signals," B.S.T.J., *36,* No. 3 (May 1957), pp. 653–709.
6. J. W. Tukey, "Nonlinear (Nonsuperposable) Methods for Smoothing Data," EASCON Record (1974), p. 673.
7. R. Steele and D. J. Goodman, "Detection and Selective Smoothing of Transmission Errors in Linear PCM," B.S.T.J., *56,* No. 3 (March 1977), pp. 399–409.
8. J. M. Tribolet, P. Noll, B. J. McDermott, and R. E. Crochiere, "A Comparison of the Performance of Four Low-Bit-Rate Speech Waveform Coders," B.S.T.J., *58,* No. 3 (March 1979), pp. 699–712.
9. D. J. Goodman, C. Scagliola, R. E. Crochiere, L. R. Rabiner, and J. Goodman, "Objective and Subjective Performance of Tandem Connections of Waveform Coders with an LPC Vocoder," B.S.T.J., *58,* No. 3 (March 1979), pp. 601–29.
10. C. S. Xydeas, C. C. Evci, and R. Steele, "Sequential Adaptive Predictors for ADPCM Speech Encoders," IEEE Trans. Commun., *COM-30,* No. 8 (August 1982), pp. 1942–54.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

On the Use of Energy in LPC-Based Recognition of Isolated Words

By M. K. BROWN and L. R. RABINER

(Manuscript received May 17, 1982)

Recognition of isolated words by encoding speech into linear predictive coefficients (LPC) is well known and widely accepted as one of the better methods for speech recognition. One of the drawbacks in relying entirely on LPC measures for recognition, however, is that the energy information in the speech is removed during the LPC analysis. Consequently, attempts have been made to include energy pattern information along with the LPC pattern information to achieve greater recognition accuracy. This paper discusses problems involved in combining energy pattern information with the LPC pattern information and presents results of recognition experiments with one method. The energy information and LPC information are combined linearly in a (speech) frame-by-frame manner utilizing the dynamic time warping (DTW) method time alignment. The LPC log likelihood ratio distance function, which determines the spectral difference between two frames of speech, does not lend itself to direct statistical analysis in multiple dimensions. The method for obtaining the weighting for the linear combination involves an iterative minimization of a probability of error function. The combined energy and LPC distance function was tested using a 129-word "airline" vocabulary, which is designed for speaker-independent, isolated word recognition. The inclusion of energy information in the recognition feature space reduces recognition error rates by an average of about 25 percent as compared with LPC alone.

I. INTRODUCTION

In the last few years it has become common to use LPC coding techniques for speech recognition.[1-7] The speech to be represented is modeled by a linear digital filter with coefficients chosen so that the transfer function of the filter approximates the spectrum of the speech over some short interval of time. Typically, a speech recognition system performs its task by comparing the unknown utterance or test with a number of previously stored reference patterns. Both the test and reference are characterized by a set of linear predictive coefficients. This is accomplished by digitizing the speech at some suitable rate and breaking the utterance into time windowed regions or "frames" upon which LPC analysis is performed. The frames of speech generally overlap and are typically spaced 10 to 20 ms apart in time. Thus, a typical 0.6-second utterance is represented by about 40 frames of linear predictive coefficients.

It is well known in the area of speech recognition that optimal time alignment of reference patterns to test patterns substantially reduces recognition errors for a vocabulary with polysyllabic words.[1] The most commonly used time alignment procedures, for the speech recognition problem, are the class of algorithms referred to as dynamic programming (DP) or dynamic time warping (DTW) methods.[2-5]

Let us assume that we are given a characterization of an isolated word that consists of a set of N vectors of LPC coefficients. The test pattern, T, is represented as:

$$\mathbf{T} = \{T(1), T(2), \cdots, T(N)\}, \tag{1}$$

where the vector $T(i)$ is a spectral (LPC) representation of the ith frame of the test word. In our system a set of nine autocorrelations constitutes the vector from which an 8th order LPC model is derived. The duration of the test utterance is N frames, where each frame represents 45 ms of speech, and adjacent frames are spaced 15 ms apart.

For a given vocabulary of V words, the reference R_v, is represented as:

$$\mathbf{R}_v = \{R_v(1), R_v(2), \cdots, R_v(M_v)\}, \tag{2}$$

where each vector, $R_v(i)$, is again a spectral representation of the corresponding frame within the reference pattern, and M_v is the number of frames in the vth reference.

To optimally align the time scale of the reference pattern (the dependent m index) to the test pattern (the independent n index), we must solve for a warping path function of the form:

$$m = w(n) \tag{3}$$

and thereby seek to minimize the total distance

$$D_v = \sum_{n=1}^{N} d\{T(n), R_v[w(n)]\} \tag{4}$$

over all possible paths, $w(n)$, where $d[T(n), R_v(m)]$ is the local distance between test frame n and reference frame $m = w(n)$. This operation must be performed for each reference R_v in the vocabulary. The test pattern is classified as belonging to the class (i.e., the reference word) for which the smallest accumulated DTW distance, D_v, is obtained. In addition to the standard DTW algorithm, time normalization has been used on both the test and reference patterns, thereby allowing the widest range of time alignment paths to be considered. This procedure, called the normalize-and-warp method,[5] linearly normalizes the test and reference utterances to a fixed length (typically the average duration of all words in the vocabulary) before the DTW is performed. Experimental results have shown this method to be valid on several recognition vocabularies, including the one used in this study.[5]

Comparison of a test frame to a reference frame requires a measure of closeness (distance). Several distance measures have been investigated and used for utterance comparison purposes.[8] Virtually all of these distance measures are spectral in nature and generally do not explicitly consider the energy pattern of the speech. The LPC-based distance measure developed by Itakura[2] has been found to yield high recognition accuracy and cost relatively little in computation. This distance function, often referred to as the log likelihood ratio (LLR) yields numerical values that are indicative of the spectral energy difference between the two frames of speech. The form of the function is as follows:

$$d(T, R) = \log[(a_R V_T a'_R)/(a_T V_T a'_T)], \tag{5}$$

where T refers to a test frame and R refers to a reference frame, a is a vector consisting of the $(p + 1)$ LPC coefficients of a pth order LPC model of the speech, and V_T is the $(p + 1 \times p + 1)$ autocorrelation matrix of the test frame. [Itakura[2] has shown how the computation of eq. (5) can be performed with $(p + 1)$ multiplication and additions and one logarithm.] The LPC coefficients do not contain any energy information since they are derived from a normalized spectrum. The V matrix enters into both the numerator and denominator of the distance function (which is the ratio of two scalars), and thus contributes no energy information. Hence, $d(T, R)$ contains no energy information.

A few further observations about the LLR distance are in order. The distance function, $d(T, R)$, does not satisfy commutative or triangle inequality rules, (i.e., the function is not symmetric). The log likelihood ratio distance is related to the coefficient sensitivities of the LPC filter model of the test utterance. If the test filter model is very similar to the reference filter model, then it is reasonable to estimate the difference between the two filter models based on the test filter coefficient sensitivities. However, if the two models differ greatly, then the coefficient sensitivities of the reference model will be much different from those of the test model and comparison of the two models yields inconsistent results. Thus, $d(T, R)$ is not monotonic when it exceeds certain values (usually about 0.6 per frame); however, it is quite useful in measuring spectral closeness (as opposed to spectral separation), and in this application serves very well for recognition purposes.

Since the log likelihood ratio distance measure normalizes energy out of the measurement, it is desirable to consider including this additional information in the distance calculation. In many pattern recognition disciplines, the addition of dimensions to a feature space is all that is normally required to add information to the distance measurement. Ordinarily, if the individual components of the distance measurement are available, an optimal weighting of features (LPC and energy) can be obtained by an analysis of feature covariance.[9,10] However, in speech recognition, there are complications that make simple addition of a dimension to the feature space difficult. One of these complications arises from the nature of the log likelihood ratio computation, which does not allow separation of the individual components of distance, i.e., $d(T, R)$ is a ratio of scalars and becomes meaningless if only one dimension of the LPC space is considered. DTW methods further complicate the addition of energy information to the feature space since the DTW path will be altered by the distances calculated during the DTW optimization. Thus, the addition of another dimension (frame energy) to the feature space is not trivial.

In the next section of this paper we will describe a new discriminant function that contains both spectral and energy information. A method for determining the weighting of the two components based on probability of recognition error will be derived. In Section III we discuss experimental results using this procedure on a vocabulary of 129 words of the "airlines" vocabulary described in Ref. 6.

II. ADDITION OF FEATURES TO THE LPC SPACE

To add the speech energy pattern information to the LPC feature space, the LPC part must be handled as a single dimension because of the log likelihood ratio distance function. Thus, if the total feature vector is otherwise treated as a linear combination of vectors, the total feature vector is of the form:

$$F = [\mathrm{LPC}, E]', \tag{6}$$

where the feature vector, F, consists of a vector of autocorrelation coefficients (treated as scalar), and a value for peak normalized log energy.

The distance function chosen for comparing the test frame energy pattern with the reference frame energy pattern, referred to as peak normalized log energy ratio, is of the form

$$e(T, R) = |\log[E(T)E_{\max}(R)/E(R)E_{\max}(T)]|, \tag{7}$$

where $E(T)$ is the energy of a test frame and $E(R)$ is the energy of a reference frame, $E_{\max}(T)$ is the peak energy of the test utterance over frames $i = 1, 2, \cdots, N$, and $E_{\max}(R)$ is the peak energy of the reference utterance over frames $j = 1, 2, \cdots, M$. This is equivalent to a peak normalized log energy difference, i.e.,

$$e(T, R) = |NE(T) - NE(R)|, \tag{8}$$

where $NE(T)$ is normalized energy and

$$NE(T) = \log[E(T)] - \log[E_{\max}(T)]. \tag{9}$$

Then for the optimal linear classifier, distance is given by

$$d(T, R) = D(T, R)'\mathbf{W}\, D(T, R), \tag{10}$$

where (lower case indicates a scalar quantity)

$$D(T, R) = F(T) - F(R) \tag{11}$$

and

$$\mathbf{W} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \tag{12}$$

where a, b, and c are chosen to minimize the probability of recognition error, $P(E)$. Applying ordinary Bayesian techniques would result in the well-known Mahalonobis distance where the matrix $\mathbf{W}$ is the inverse covariance matrix of the feature space.[9,10] Because of the nature of the log likelihood ratio distance function, the LPC distance to the origin of the LPC space is generally too large to be within the 0.6 value required for monotonicity. Furthermore, no other point in the LPC space can easily be found that will allow this requirement to be satisfied for all possible samples in the space. Hence, the mean and, in turn, variance (and, hence, covariance) of the LPC component cannot be determined directly.

An alternate method of combining the energy measure with LPC has been developed which, although not able to determine the cross-product coefficients, will determine a weighting of the two measures based on probability of error. Let the distance function assume the form:

$$d(T, R) = [\mathrm{LLR}(T, R)] + \alpha[\mathrm{LER}(T, R)], \tag{13}$$

where $\mathrm{LLR}(T, R)$ is the log likelihood distance between a test frame T and a reference frame R, and $\mathrm{LER}(T, R)$ is the peak normalized log energy difference between test frame T and reference frame R, and α is a weighting coefficient. Equation (13) must be employed in (4) to determine the DTW function (3) and choose the closest reference to a given test utterance. Consequently, the LLR distance is a function of the LER distance and vice versa, since the DTW path is a function of both distances.

Let $D(i, j)$ indicate the accumulated DTW distance between a test utterance (corresponding to word i) and reference pattern (corresponding to word j). A classification error will occur when $D(i, i) > D(i, j)$ for *any* j not equal to i. That is, if the distance between a test class (i) and a reference of the same class (i) is greater than the distance from class (i) to a different class (j), then a recognition error will occur. An alternate form of (13) and (14) is the discriminant function:

$$Q(i, j) = D(i, j) - D(i, i). \tag{14}$$

For this form a recognition error occurs only if $Q(i, j)$ is less than zero for any j not equal to i.

For notational convenience, let

$$L(i, j) = \mathrm{LLR}(i, j) - \mathrm{LLR}(i, i) \tag{15}$$

and

$$E(i, j) = \mathrm{LER}(i, j) - \mathrm{LER}(i, i), \tag{16}$$

where LLR(i, j) and LER(i, j) are the accumulated log likelihood ratio and log energy ratio on the DTW path, respectively. Then

$$Q(i, j) = L(i, j) + \alpha E(i, j). \tag{17}$$

There are four kinds of classification errors for which $Q(i, j)$ is less than zero and a test word will be misclassified, namely:

(A) an LLR error for which $L(i, j) < 0$ for any $j \neq i$ and $E(i, j) > 0$ for all $j \neq i$

(AB) an LLR error or an LER error but not both, i.e.,
 (a) $L(i, j) < 0$ and $E(i, j) > 0$ for any $j \neq i$
 (b) $L(i, k) > 0$ and $E(i, k) < 0$ for any $k \neq i, j$

(B) an LER error for which $L(i, j) > 0$ for all $i \neq j$ and $E(i, j) < 0$ for any $j \neq i$

(C) both errors for which $L(i, j) < 0$ and $E(i, j) < 0$ for any $j \neq i$.

The test word is correctly recognized when condition (A) exists if $Q(i, j) > 0$ for all $j \neq i$, which implies:

$$\alpha > |L(i, j)/E(i, j)| \quad \text{for all} \quad j \neq i. \tag{18}$$

Likewise, the test word is correctly recognized when condition (B) exists if:

$$\alpha < |L(i, j)/E(i, j)| \quad \text{for all} \quad j \neq i. \tag{19}$$

Condition (C) is not recoverable [i.e., $Q(i, j) < 0$] for any value of α.

In this case the test word will always be misrecognized since both the LLR and LER distances have made an error.

Condition (AB) is a special case where an LPC-type error $[L(i, j) < 0]$ occurs for a comparison of the test (i) with one reference (j) and an LER-type error $[E(i, k) < 0]$ occurs for a comparison with a different reference (k). In order for this test word to be properly recognized the following relation must be satisfied for all $j, k \neq i$:

$$|L(i, j)/E(i, j)| < \alpha < |L(i, k)/E(i, k)|. \tag{20}$$

Equation (20) may or may not be satisfiable for a given test word (i).

The probability of recognition error can be written with the further definition of the random variable:

$$x = |L(i, j)/E(i, j)|. \tag{21}$$

Define the probability density function of x conditioned on error type (A) as $p(x|A)$ and the probability density function of x conditioned on error type (B) as $p(x|B)$. Then the probability of error, $P(E, \alpha)$, is given by:

$$P(E, \alpha) = P(A) \int_{\alpha}^{\infty} p(x|A)dx + P(B) \int_{-\infty}^{\alpha} p(x|B)dx + P(C)$$
$$+ P(AB) \int_{\alpha}^{\infty} p(x|AB, a)dx + P(AB) \int_{-\infty}^{\alpha} p(x|AB, b)dx, \tag{22}$$

where $p(x|AB, a)$ is the probability density function conditioned on the existence of error type (AB) of the first kind $[L(i, j) < 0$ and $E(i, j) > 0]$ and $p(x|AB, b)$ is the density function conditioned on error type (AB) of the second kind $(L(i, k) > 0$ and $E(i, k) < 0)$. Optimizing (22) with respect to α yields the relation:

$$P(A)p(x = \alpha^*|A) + P(AB)p(x = \alpha^*|AB, a)$$
$$= P(B)p(x = \alpha^*|B) + P(AB)p(x = \alpha^*|AB, b), \tag{23}$$

where α^* is the optimal value of α. Inspection of (22), (23), and the error conditions (A), (AB, a), (B), and (AB, b) reveals that they are mutually exclusive events and that they can be combined. Thus, by relaxing the requirements "for all $j \neq i$" of condition (A) and (B) to "for any $j \neq i$" eq. (23) may be rewritten

$$P(A)p(x|A) = P(B)p(x|B), \tag{24}$$

keeping in mind the new properties of (A) and (B):

$$(A)\ L(i, j) < 0 \quad \text{and} \quad E(i, j) > 0 \quad \text{for any} \quad j \neq i$$
$$(B)\ L(i, j) > 0 \quad \text{and} \quad E(i, j) < 0 \quad \text{for any} \quad j \neq i.$$

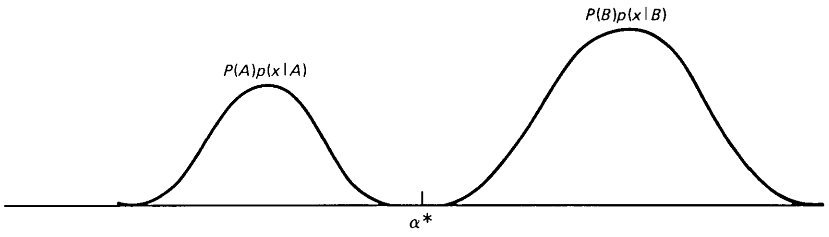


Fig. 1—Conditions for minimum-recognition error rate.

Several observations concerning (18) through (24) are worth noting. First, (22) indicates that the best performance that can be expected for any choice of α is

$$P(E, \alpha^*) = P(C), \tag{25}$$

where α^* is the value of α for which $P(E, \alpha)$ is minimized. This condition occurs only if α can be chosen so that the distribution $P(A)p(x|A)$ lies entirely below α and $P(B)p(x|B)$ lies entirely above α, as shown in Fig. 1. If a recognizer is implemented with only the LLR distance function, the probability of error will be

$$P(E|\text{LLR only}) = P(A) + P(C) \tag{26}$$

since error conditions (A) and (C) are mutually exclusive LLR-type errors. Hence, if $P(A) > 0$, then

$$P(C) < P(E|\text{LLR only}), \tag{27}$$

indicating with (25) that

$$P(E, \alpha^*) < P(E|\text{LLR only}) \quad \text{if} \quad P(A) > 0 \tag{28}$$

and therefore recognition performance better than that obtained with LPC alone can be achieved by using distance function (13). Note also by (21) that random variable $x \geq 0$. The lower limits of the second and fourth integrals in (22) can be made 0 without effect on $P(E, \alpha)$. Then, under the worst-case condition that $P(A) = 0$, the optimal value for α is $\alpha^* = 0$. Inspection of (13) indicates that, for $\alpha^* = 0$, $D(i, j)$ is identical to the LPC component of the distance function, LLR(i, j). Thus, the worst performance that can be expected is the LPC error rate.

Under good conditions, $|L(i, j)|$ will be small for condition (A) and $|E(i, j)|$ will be small for condition (B). Thus, the mean of the distribution $P(A)p(x|A)$ will be low, while the mean of the distribution $P(B)p(x|B)$ will be large, as shown in Fig. 2. The probability of recognition error, which consists of $P(C)$ plus the shaded area shown in Fig. 2, is minimized by the value of α^* shown. If we assume that $P(A)p(x|A)$ and $P(B)p(x|B)$ are normally distributed as in Fig. 2, then the shaded area will be less than the area of $P(A)p(x|A)$, i.e.,

$$P(E, \alpha^*) < P(A) + P(C), \tag{29}$$

which by (26) yields

$$P(E, \alpha^*) < P(E|\text{LLR only}). \tag{30}$$

Thus, not only can we guarantee that performance will be no worse than LPC recognition performance, but, if the conditions of Fig. 2 can be established, we can guarantee better performance by properly choosing α.

Differing DTW paths owing to the interaction of the LLR and LER components of $d(T, R)$ will cause the distributions $P(A)p(x|A)$ and $P(B)p(x|B)$ to vary with different values for coefficient α. Consequently, the determination of coefficient α may require several iterations of selecting α and redefining (13) until a stable value for α is obtained. Under adverse conditions $P(A)p(x|A)$ and $P(B)p(x|B)$ may overlap considerably and the number of recoverable errors may be small.

III. EXPERIMENTAL RESULTS

Experiments were conducted on speech collected from several talkers speaking an "airlines" vocabulary.[6] This vocabulary consisted of 129 words that would commonly be used to obtain information from airline scheduling service. The range of words is broad enough to be considered a good cross section of English speech. The reference words were generated by clustering the speech of several male and female talkers to form speaker-independent templates.[7] Six clustered templates for each of the 129 words were generated resulting in a reference file of 774 templates. The average duration of all words was 42 frames and this was the duration used for word length normalization.

Test sets were studied for two male and two female talkers (not of the reference set) in order to choose an approximate value for the weighting coefficient α. The LLR and LER distance functions were not linked together for these experiments. Thus, two different DTW paths were generated, one for LLR distances and one for LER dis-

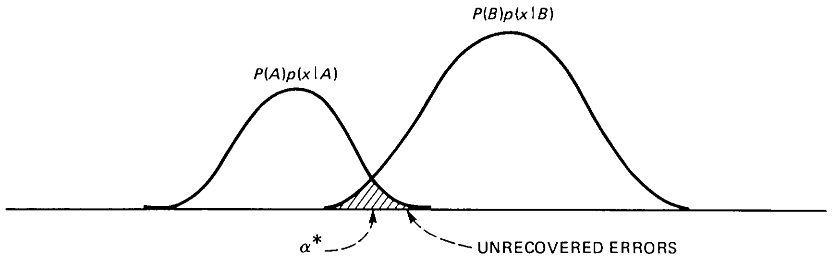


Fig. 2—Conditions for low-recognition error rate.

Table I—Statistics on LLR and LER distance measures for the four data sets

Data Set (Talker Sex)		1 (M)	2 (M)	3 (F)	4 (F)
LLR Recognition Error	First Choice	9.3%	12.4%	22.5%	21.7%
	First Two Choices	3.9%	5.4%	10.8%	12.4%
LER Recognition Error	First Choice	77.5%	72.1%	75.2%	73.6%
	First Two Choices	66.7%	60.5%	63.6%	58.9%
Correlation LLR vs. LER	Correct Words	0.093	0.183	0.205	0.227
	Incorrect Words	0.205	0.175	0.143	0.144

tances. The accumulated LER distance was also determined along the DTW path generated by the LLR distances to evaluate the effects of the warp path on the accumulated LER distance. (It was found, early in the investigation, that the LLR distance would be the dominant force in directing the DTW path.) Statistics were gathered on the number of recognition errors for both distance measures, the distributions for LLR distances and LER distances were calculated, and the distributions $P(A)p(x|A)$ and $P(B)p(x|B)$ were determined. In addition, correlation matrices were calculated for LLR, LER, and LER along the LLR path. The results are shown in Table I and Fig. 3. The

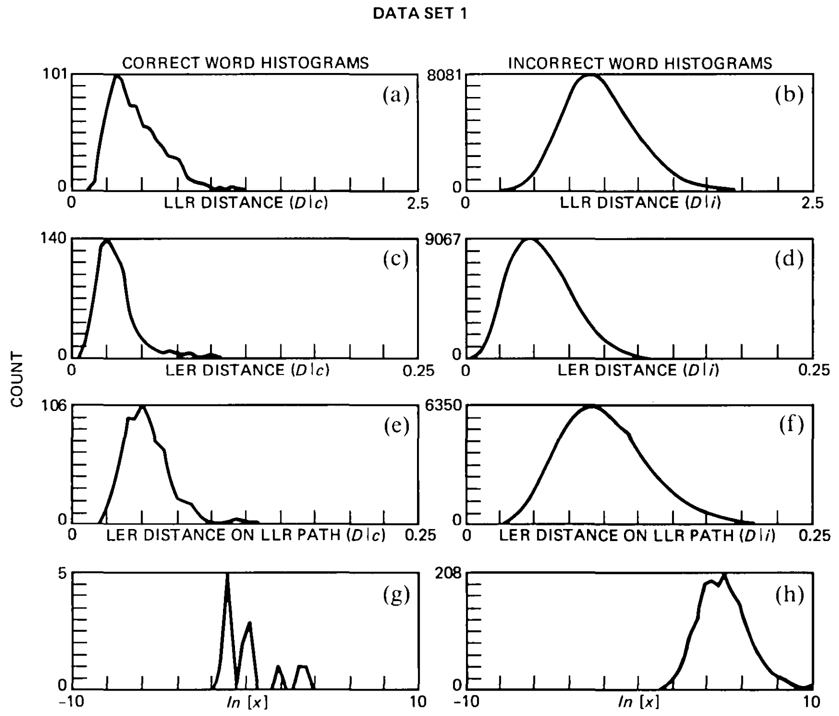


Fig. 3—DTW statistics for unlinked LLR and LER distances.

captions in Fig. 3 are read as follows. $D|c$ is distance to correct reference, $D|i$ is distance to incorrect reference, and LER(LLR) is the LER distance along the LLR-controlled DTW path.

The plots of $P(A)p(x|A)$ and $P(B)p(x|B)$ of Fig. 3 indicate that an initial value of α should be between 1.0 and 5.0. The correlations of LLR and LER (Table I) indicate that there is little redundancy and, hence, useful information may be obtainable by combining the LLR and LER distance measures. The plots of $P(A)p(x|A)$, as shown in the figure, are quite jagged due to the low number of samples obtained; however, the optimal value of α was actually determined by inspection of the samples of the two distributions. Although this is a somewhat tedious process, it ensures that the best value for α is obtained from the available information. The graphical display of $P(A)p(x|A)$ and $P(B)p(x|B)$ gives good indication that the requirements for (29) and (30) are being met.

Table I indicates that the recognition error rate for LER distance measures alone is very high. Even among the top two candidates, the correct word is found less than half of the time. Obviously, LER alone is not a good discriminating feature. The average LER distance [eq. (7)] from the test to a correct reference (i.e., from the same class as the test) is low in comparison to the LER distance to an incorrect class (about 1:1.75). This would normally indicate a good discriminating feature; however, the LER distances to both correct and incorrect references are very widely distributed and the amount of overlap of the two distributions is considerable (see Fig. 3). The LER distances along the DTW path of the LLR measure are less widely distributed but the overlap is still large. For comparison, the ratio of test to correct reference LLR distance with respect to incorrect reference LLR distance averages about 1:1.97, and the amount of overlap of the distributions is relatively small.

Tests were next conducted on the same speech data with the LLR and LER distances linked for DTW path determination. An initial value of $\alpha = 1.0$ was used as a starting point for each test set. Statistics were gathered as previously, except that the combined LLR and LER distance, LLR component, and LER component were items of interest. The distributions of $P(A)p(x|A)$ and $P(B)p(x|B)$ were again calculated.

Several iterations of testing and evaluating were performed to obtain a good estimate for the coefficient α^*. For speech data set 1 the value obtained for α^* was 1.8. The statistical results are shown in Fig. 4 and Table II. A plot of the number of recognition errors predicted by the distributions of $P(A)p(x|A)$ and $P(B)p(x|B)$ as a function of x is shown in Fig. 5a for $\alpha = 3.0$. The same test was run on the second set of speech data (at $\alpha = 3.0$), and the plot shown in Figure 5b was

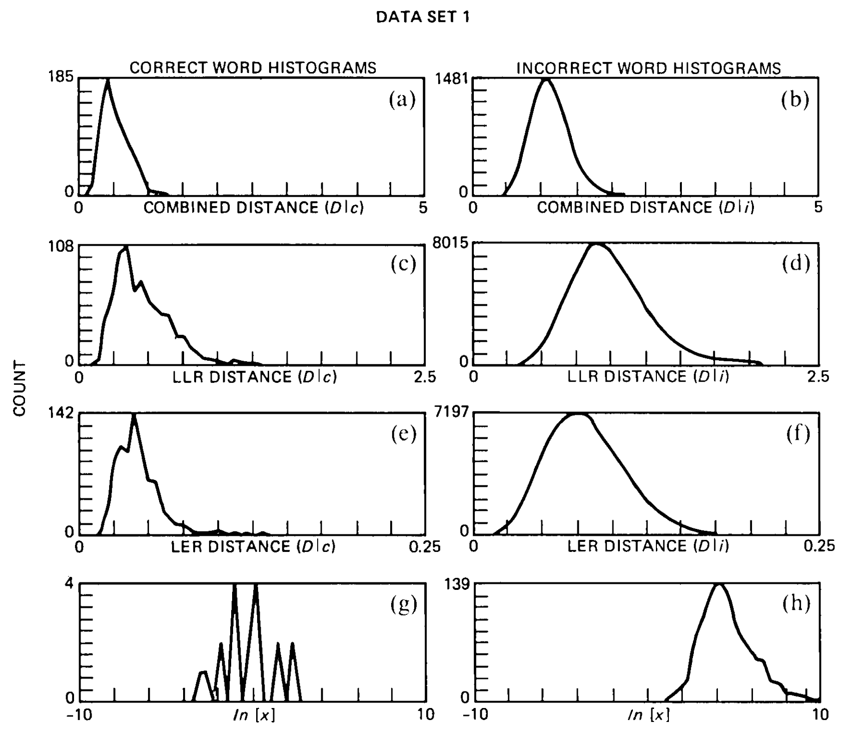


Fig. 4—DTW statistics for linked LLR and LER distances.

obtained. Iterative application of the evaluation procedure yielded a final value of $\alpha = 5.0$ for test set 2. Obviously, the selection of α is very sensitive to the speech data. In all cases, a value for α^* could be found to improve the recognition accuracy over that for LPC-based recognition alone (indeed, for $\alpha = 3.0$ the recognition error rate was lower for both data sets 1 and 2) but the optimal values predicted by error rate plots like those of Fig. 5 were considerably different.

Further testing indicated that a single value for α could be chosen so that an improvement in recognition performance would be obtained for all of the testing sets. The values of $\alpha = 2.0$ and $\alpha = 3.0$, as indicated in Table II, show significant improvement for all four test sets. The average reduction in error rate is 23.9 percent for $\alpha = 2.0$ and 29.8 percent for $\alpha = 3.0$. Thus, a speaker-independent recognizer can make use of the improvement in performance available by the inclusion of energy information using this technique.

3.1 Analysis of the recognition errors

It is interesting to examine the specific errors that were connected by using the combined energy plus LPC feature set. A list of all such

Table II—Word error rates as a function of α for the four data sets

Data Set (Talker Sex)		1 (M) (%)	2 (M) (%)	3 (F) (%)	4 (F) (%)
$\alpha = 0.0$*	First Choice	9.3	12.4	22.5	21.7
	First Two	3.9	5.4	10.8	12.4
$\alpha = 1.0$	First Choice	6.2	10.8	17.8	20.9
	First Two	3.1	5.4	8.5	8.5
$\alpha = 1.8$	First Choice	3.1	10.1	...	...
	First Two	2.3	7.0	...	...
$\alpha = 2.0$	First Choice	3.9	10.8	20.2	18.6
	First Two	2.3	7.0	9.3	9.3
$\alpha = 2.2$	First Choice	...	...	...	18.6
	First Two	...	...	...	9.3
$\alpha = 2.8$	First Choice	...	...	18.6	...
	First Two	...	...	7.7	...
$\alpha = 3.0$	First Choice	3.9	10.1	17.8	17.0
	First Two	2.3	7.0	7.7	10.8
$\alpha = 3.4$	First Choice	...	...	17.0	...
	First Two	...	...	9.3	...
$\alpha = 3.7$	First Choice	...	...	...	18.6
	First Two	...	...	...	10.8
$\alpha = 5.0$	First Choice	...	9.3	...	...
	First Two	...	5.4	...	...
$\alpha = 5.5$	First Choice	...	10.1	...	...
	First Two	...	5.4	...	...

* Equivalent to LLR only.

errors is given in Table III. This table gives the correct word, the word recognized using LPC alone ($\alpha^* = 0$), the test set in which the error occurred, and a classification as to the type of error initially made. The classification code describes the phonetic nature of the correct and misrecognized words as one of the following:

(*i*) MS—Simple monosyllabic word

(*ii*) MS + A—monosyllabic word plus affix (final stop or fricative consonant)

(*iii*) PS—polysyllabic word.

An examination of the 34 words in Table III shows that 11 of the corrections involved a polysyllabic word (as either the correct word or the word recognized using LPC alone), nine of the corrections involved monosyllabic words with affixes, and the remaining 14 corrections involved monosyllabic words.

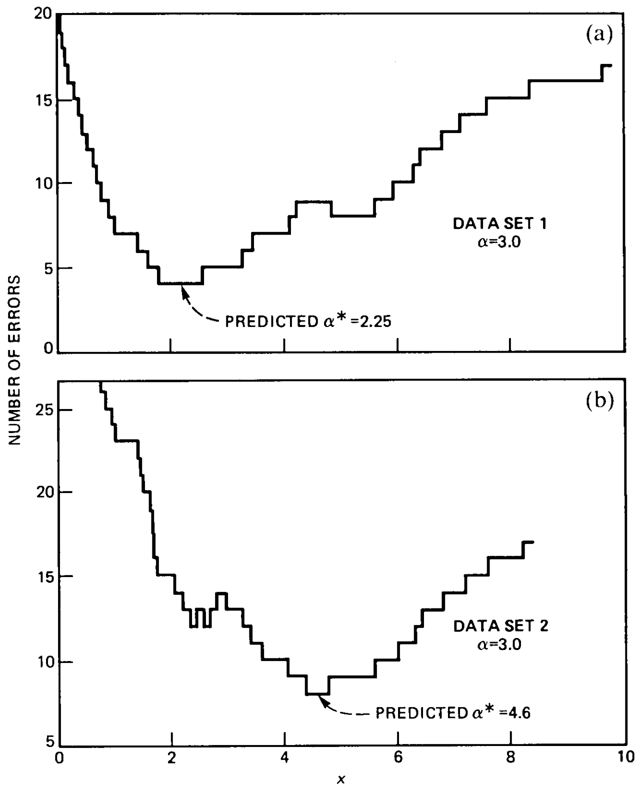


Fig. 5—Prediction of α from statistics on x.

A similar list of the words that were incorrectly recognized using the combined distance metric is given in Table IV. The format for this table is similar to that of Table III except that the classification code refers to the correct word and the word originally recognized using the LPC distance alone. It can be seen that eight new errors are introduced by the combined metric that were not present using LPC alone. Hence, a total net improvement of 26 words was obtained using the combined distance metric.

Of the 53 errors given in Table IV, 16 involve a polysyllabic word (either the correct or the LPC misrecognized word), 23 involve only monosyllabic words, and 13 involve polysyllabic words with affixes.

The data of Tables III and IV indicate that the inclusion of energy into the distance metric leads to a fairly uniform improvement in accuracy across all three types of word classifications. The results also indicate that, for the most part, the remaining errors involve acoustically similar words, whereas the corrections generally come from errors involving acoustically different sounding words.

Table III—List of words misrecognized using LPC alone but correctly recognized using energy combined with LPC

Correct Word	Word Recognized From LPC Alone	Test Set	Classification Code
Boston	Washington	3	(PS, PS)
Card	I	3	(MS+A, MS)
Depart	July	4	(PS, PS)
Detroit	Information	1	(PS, PS)
Does	Five	3	(MS, MS)
Eight	Seat	1	(MS+A, MS+A)
First	Express	3	(MS+A, PS)
First	Express	4	(MS+A, PS)
I	Lockheed	3	(MS, PS)
Like	Flight	4	(MS+A, MS+A)
Many	May	3	(MS, MS)
May	Change	2	(MS, MS)
Morning	Miami	4	(PS, PS)
My	By	2	(MS, MS)
Number	November	4	(PS, PS)
Oh	Of	4	(MS, MS)
On	Arrival	4	(MS, PS)
One	What	3	(MS, MS+A)
Pay	A	2	(MS, MS)
Phone	Five	2	(MS, MS)
Please	Seat	1	(MS, MS+A)
Seats	Seat	1	(MS+A, MS+A)
Seats	Seat	3	(MS+A, MS+A)
Some	From	3	(MS, MS)
Some	From	4	(MS, MS)
Ten	Afternoon	4	(MS, PS)
There	Fare	3	(MS, MS)
Three	Detroit	2	(MS, PS)
Time	Card	3	(MS, MS+A)
Times	Five	1	(MS+A, MS)
To	Do	3	(MS, MS)
Twelve	Five	4	(MS, MS)
Uh	How	3	(MS, MS)
When	Would	1	(MS, MS)

IV. CONCLUSIONS

The addition of energy information to the LPC distance improves recognition performance significantly. It is likely that the energy information would significantly improve the error rate on certain kinds of anomalies in the speech, such as partially voiced words and lip pops. These anomalies sometimes cause a test word to match reference words that have similar spectral patterns but significantly different energy patterns. Unfortunately, data bases of speech containing these kinds of erroneous sounds are not yet readily available and testing of this hypothesis must be deferred until such data are generated.

The selection of the weighting coefficient, α, is quite sensitive to the speech data. For that reason, the value of α^* will usually be different

Table IV—Word incorrectly recognized using the combined distance metric

Correct Word	Word Recognized From LPC Alone	Word Recognized From LPC + Energy	Test Set	Classification Code
A	May	Pay	2	(MS, MS)
A	*A*	Pay	3	(MS, MS)
Boston	*Boston*	What	2	(PS, MS+A)
By	I	I	2	(MS, MS)
By	I	I	3	(MS, MS)
By	Like	I	4	(MS, MS+A)
Change	*Change*	Stops	4	(MS, MS+A)
Code	Card	Card	3	(MS+A, MS+A)
Code	Card	Card	4	(MS+A, MS+A)
Code	Coach	Coach	2	(MS+A, MS)
DC	BAC	BAC	4	(PS, PS)
Do	Will	Code	4	(MS, MS)
Eight	*Eight*	Take	3	(MS+A, MS+A)
Flight	*Flight*	Flights	4	(MS+A, MS+A)
Flights	Flight	Flight	3	(MS+A, MS+A)
Flights	Flight	Flight	4	(MS+A, MS+A)
For	Five	Five	4	(MS, MS)
Four	Five	Five	4	(MS, MS)
From	*From*	Eleven	4	(MS, PS)
Go	Twelve	Club	3	(MS, MS)
Home	How	How	4	(MS, MS)
I	By	By	1	(MS, MS)
In	*In*	AM	3	(MS, PS)
Is	In	In	4	(MS, MS)
Leave	Please	Please	3	(MS, MS)
Many	Pay	Pay	1	(PS, MS)
March	Much	Much	4	(MS, MS)
Much	March	March	3	(MS, MS)
My	I	By	3	(MS, MS)
Nine	Washington	Morning	4	(MS, PS)
October	September	September	1	(PS, PS)
Oh	How	How	2	(MS, MS)
Oh	How	How	3	(MS, MS)
On	Five	Five	2	(MS, MS)
One	When	When	4	(MS, MS)
PM	Seattle	Seattle	4	(PS, PS)
Pay	Friday	Friday	3	(MS, PS)
Phone	From	From	4	(MS, MS)
Please	April	Thee	4	(MS, PS)
Prefer	There	Fare	3	(PS, MS)
Seat	Seats	Seats	3	(MS+A, MS+A)
Sunday	Saturday	Saturday	3	(PS, PS)
Ten	AM	PM	2	(MS, PS)
The	Five	Five	3	(MS, MS)
Thee	DC	DC	1	(MS, PS)
Thee	Make	Three	2	(MS, MS+A)
There	Fare	Fare	2	(MS, MS)
Times	Office	Five	3	(MS, PS)
Two	Do	Do	4	(MS, MS)
Want	What	What	4	(MS+A, MS+A)
What	Want	Want	2	(MS+A, MS+A)
When	*When*	Morning	4	(MS, PS)

for each talker in the testing set. Consequently, this method is likely to be more effective for speaker-trained recognition than for speaker-independent recognition systems. However, a value of $\alpha = 2$ or $\alpha = 3$ has been found to work well for the test sets used in this study and will probably work for most test sets.

REFERENCES

1. G. M. White and R. B. Neely, "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-24*, No. 2 (April 1976), pp. 183–8.
2. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-23*, No. 1 (February 1975), pp. 67–72.
3. H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-26*, No. 1 (February 1978), pp. 43–9.
4. L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson, "Considerations in Dynamic Time Warping for Discrete Word Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-26*, No. 6 (December 1978), pp. 575–82.
5. C. S. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-28*, No. 6 (December 1980), pp. 622–33.
6. J. G. Wilpon, L. R. Rabiner, and A. Bergh, "Speaker Independent Isolated Word Recognition Using a 129 Word Airline Vocabulary," J. Acoustical Society of America, *72,* No. 2 (August 1982), pp. 390–6.
7. L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker Independent Recognition for Isolated Words Using Clustering Techniques," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-27*, No. 4 (August 1979), pp. 336–49.
8. A. H. Gray and J. D. Markel, "Distance Measures for Speech Processing," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-24*, No. 5 (October 1976), pp. 380–91.
9. G. S. Sebestyen, *Decision Making Processes in Pattern Recognition*, New York: Macmillan, 1962.
10. N. J. Nilsson, *Learning Machines: Foundations of Trainable Pattern Classifying Systems*, New York: McGraw-Hill, 1965.
11. J. M. Tribolet and L. R. Rabiner, "Statistical Properites of an LPC Distance Measure," IEEE Trans. on Acoustics, Speech, and Signal Processing, *ASSP-27*, No. 5 (October 1979), pp. 550–8.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Isolated Word Recognition for Large Vocabularies

By L. R. RABINER, A. E. ROSENBERG, J. G. WILPON, and W. J. KEILIN*

(Manuscript received June 2, 1982)

It has long been known that one of the key factors in determining the accuracy of isolated word recognition systems is the size and/or complexity of the vocabulary. Although most practical isolated word recognizers use small vocabularies (on the order of 10 to 50 words), there are many applications that require medium- to large-size vocabularies (e.g., airlines reservation and information, data retrieval, etc). This paper discusses the problems associated with speaker-trained recognition of a large vocabulary (1109 words) of words. It is shown that the practicability of using large vocabularies for isolated word vocabularies is doubtful, both because of the problems in training the system, and because of the difficulty the user has in learning and remembering the vocabulary words for any significant size vocabulary. The importance of studying large word vocabularies for recognition lies in the flexibility it provides for understanding the effects of vocabulary size and complexity on recognition accuracy for both small- and medium-size vocabularies. By constructing subsets of the total vocabulary for recognition, we show that a judicious choice of words can lead to significantly better recognition accuracy than a poor choice of the words in the subset. We show that for each doubling of the size of the vocabulary, the recognition accuracy tends to decrease by a fixed amount, which is different for each talker.

I. INTRODUCTION

In the field of automatic speech recognition, the only type of system to date that has proven useful and practical is the isolated word recognizer. Isolated word recognizers have been in use commercially for a number of years,[1-5] and have been extensively studied in several

* Work performed on BLESP summer assignment at Bell Laboratories.

major research laboratories throughout the world.[6-13] For the most part, applications of isolated word recognizers have limited themselves to vocabulary sizes ranging from small (10 to 30 words) to moderate (30 to 200 words). There are several reasons why there are no commercially available systems that can recognize words from large vocabularies (greater than 200 words). These include:

(*i*) The difficulty of training the system on large vocabularies

(*ii*) The storage required for word templates for large vocabularies

(*iii*) The processing required to recognize words from large vocabularies

(*iv*) The difficulty of accurately recognizing word vocabularies that would be useful in a variety of applications.

The computational problems associated with reasons *ii* and *iii* above (i.e., large storage and large amounts of computation) are rapidly becoming less important as memory and processing costs decrease, and should continue to do so for the forseeable future. The problems in training are very real ones, and will be discussed further in this paper. The problems associated with choice of vocabulary words and accuracy of word recognition are the main topics of this paper.*

Although the practicability of large vocabularies for isolated word recognition is doubtful, the experimental use of large vocabularies provides the opportunity to examine significant issues in automatic word recognition that cannot be examined with small vocabularies. If the vocabulary is sufficiently general, in some sense, it is possible to choose several smaller partitions from the vocabulary, of a given size or complexity, and thereby better understand the effects of vocabulary size, or complexity, on word recognition accuracies.

At the present time it is not even known how currently available isolated word recognizers would perform on large vocabularies—i.e., what factors would most influence accuracy. For small- and medium-size vocabularies there is a wide body of experimental data that indicates that vocabulary complexity (not size) is the key indicator of accuracy.[8,12,14] Furthermore, most experimental studies have shown that speaker-independent word recognizers can (and do) perform as well as speaker-trained recognizers; however, they require an order of magnitude more computation.[15]

A brief summary of recent experimental results on isolated word recognition is given in Table I. The results given in this table illustrate the complex relationship between accuracy and vocabulary size and complexity. In Section II of this paper we give a simple model that helps to explain this relationship in terms of the relationships between

* For any practical system, using a large vocabulary of isolated words, syntactic constraints of the recognition task would effectively reduce the vocabulary size and speed up the processing most of the time.

Table I—Accuracies of isolated word recognizers on several vocabularies

Vocabulary	Speaker Trained (%)	Speaker Inde-pendent (%)
10 digits [1, 12]	99	98
26 letters of alphabet [16, 17]	80	70
39 alphadigits [8, 12]	87	80
54 computer terms [18, 19]	99	96
91 North American States [14]	99	—
129 Airline terms [20, 21]	88	91
561 Words and Phrases [18]	92	—

words in the vocabulary. In Section III we describe an experiment designed to measure word recognition accuracy for an 1109-word vocabulary. The recognizer was run in a speaker-trained mode on six talkers (three male, three female), in which each talker used a robust training procedure to give individual word templates. In Section IV we discuss results on recognition of subsets of the 1109-word vocabulary. These results illustrate the degree to which choice of vocabulary words can influence word accuracy for given vocabulary sizes. Finally, in Section V we summarize our findings and discuss their implications for practical systems.

II. MODEL FOR ISOLATED WORD RECOGNITION ACCURACY AND COMPLEXITY

Assume we have a specified vocabulary, V, of Q words, i.e.,

$$V = \{v_1, v_2, \cdots, v_Q\}. \tag{1}$$

We define a word similarity index as $D(v_i, v_j)$, which measures the distance (in whatever units are desirable) between pairs of vocabulary words, v_i and v_j. The distance can be an acoustic one (e.g., the average distance of the time-aligned words) or a phonetic one [e.g., the average number of phonemes (syllables, demisyllables) that are different in the words]. We next define a word overlap index, q_i, for the ith vocabulary word as

$$q_i = C\{j\text{:}s.t.\ D(v_i, v_j) \le T\}, \tag{2}$$

where C is the cardinality of the set of indices j such that the pairwise word distance score, $D(v_i, v_j)$* falls below a threshold T. Basically, q_i is a count of the number of words in the vocabulary similar to word v_i.

We can now define an average probability of error as

* For simplicity we assume that work distances, D, are symmetric, i.e., $D(v_i, v_j) = D(v_j, v_i)$. In practice, for nonsymmetric distances we use the average pairwise word distance, i.e., $[D(v_i, v_j) + D(v_j, v_i)]/2$.

$$P(E_Q) = \sum_{i=1}^{Q} P(v_i)P(E|v_i), \tag{3}$$

where $P(v_i)$ is the a priori probability word v_i is spoken, $P(E|v_i)$ is the probability of error given word v_i is spoken.[†] Since we assume all words are equiprobable, we have

$$P(v_i) = \frac{1}{Q}. \tag{4}$$

We now make the simplistic assumption that the probability of error given word v_i is spoken can be written as

$$P(E|v_i) = 1 - \frac{1}{q_i}, \tag{5}$$

i.e., we assume a random choice is made among the q_i similar versions of word v_i. Clearly the resulting error rate based on this assumption is an overbound on the true probability of error. Combining eqs. (2) through (5) we get

$$P(E_Q) = \frac{1}{Q} \sum_{i=1}^{Q} \left(1 - \frac{1}{q_i}\right). \tag{6}$$

To illustrate the interpretation of eq. (6) consider calculating the average value of q_i as

$$\bar{q} = \frac{1}{Q} \sum_{i=1}^{Q} q_i. \tag{7}$$

The quantity $\bar{q}$, which we call the average vocabulary complexity, is a measure of the average number of candidates in the vocabulary similar to any word. Since q_i satisfies the constraint

$$1 \le q_i \le Q \tag{8a}$$

then $\bar{q}$ satisfies the constraint

$$1 \le \bar{q} \le Q. \tag{8b}$$

Consider now a $Q = 10$ word vocabulary. We can define various possible sets of q_i and compute $P(E_Q)$ and $\bar{q}$ for each set. For example, if we have

$$\{q_i\} = \{3, 3, 3, 3, 3, 3, 2, 2, 2, 2\} \tag{9a}$$

then $\bar{q} = 2.6$ and $P(E_Q) = 0.6$. Similarly, if

$$\{q_i\} = \{7, 7, 7, 7, 7, 7, 7, 1, 1, 1\} \tag{9b}$$

[†] Technically, eq. (3) should contain a small residual error term that accounts for errors owing to improper recordings, mispronunciations, etc. We will omit this term for simplicity.

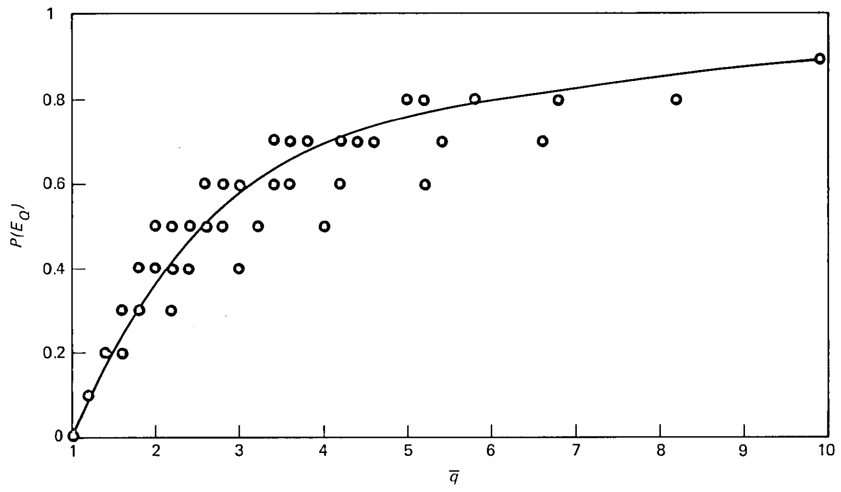


Fig. 1—Plot of average word error rate as a function of average word complexity for all possible combinations of a 10-word vocabulary. The smooth curve is a hand-drawn curve, which approximates the average behavior of the data.

then $\bar{q} = 5.2$ and $P(E_Q) = 0.6$. The vocabulary of eq. (9a) consists of four subsets, two of which have three confusable words, and two of which have two confusable words. The vocabulary of eq. (9b) consists of one subset with seven confusable words, and three distinct words. Both vocabularies, however, yield identical error probabilities using the simple model given above.

If we consider all possible subsets of the 10-word vocabulary, and plot the values of $P(E_Q)$ versus $\bar{q}$ for each such subset, the resulting plot would be as shown in Fig. 1. This figure shows that for a given probability of error a wide range of vocabulary complexities can often be found. It also shows that as the probability of error goes to the residual value, the choice of vocabularies becomes sparse—i.e., only well-designed vocabularies will achieve the lowest error rates.

This simple word recognition model could also be described in information theoretic terms based on channel models.[22] From such models one could derive plots equivalent to the one of Fig. 1.

Consider applying the word recognition model to some of the vocabularies of Table I. For the 10 digits we get (using $q_i = 1$, all i) $P(E_{10}) = 0$, $\bar{q} = 1$. For the 26 letters of the alphabet (ordered alphabetically), using*

$$\{q_i\} = \{1, 5, 2, 5, 5, 2, 5, 1, 2, 2, 2, 1, 2, 2, 1, 2, 1, 1, 2, 2, 1, 5, 1, 1, 2, 2\}$$

we get $\bar{q} = 2.2$, $P(E_{26}) = 0.385$. For the 39-word alphadigit vocabulary[16]

* The values of q_i for the alphabet were obtained from the letter confusion matrix in Ref. 16.

we get $\bar{q} = 2$ and $P(E_{39}) = 0.28$. For the 54-word computer terms vocabulary[18,19] we get $\bar{q} = 1.1$ and $P(E_{54}) = 0.04$. For the 129-word airline terms vocabulary[20,21] we get $\bar{q} \approx 1.1$ and $P(E_{129}) = 0.08$. By comparing the error probabilities from the model with those given in Table I it can be seen that a reasonable match to all vocabularies can be obtained using this simple recognition model.

The major purpose of this section has been to illustrate the range of variability in error rate associated with a vocabulary of fixed-size Q words, and to roughly explain the source of variation. The key point is to keep in mind that judicious choice of vocabulary items can lead to considerably higher word recognition accuracies than can a poor choice of vocabulary items. We will illustrate this key point further in later sections of this paper.

III. WORD RECOGNITION ON AN 1109-WORD VOCABULARY

To evaluate the performance of an isolated word recognizer on large vocabularies, the linear predictive coefficient (LPC) based recognizer developed at Bell Laboratories was tested on a vocabulary of 1109 words from the Basic English vocabulary of Ogden.[23] The recognizer was tested in a speaker-trained mode with six talkers (three male, three female) each training the recognizer.

Before presenting results of the evaluation tests, we briefly review the techniques used for recognition and training.

3.1 The LPC-based word recognizer

Figure 2 shows a block diagram of the LPC-based word recognizer. The input speech signal, $s(n)$, recorded off a standard dialed-up telephone line, is bandpass-filtered between 100 and 3200 Hz, and digitized at a 6.67-kHz rate. The first step in the processing is the preprocessing and blocking step, which consists of a simple first-order preemphasis network. The preemphasized signal is blocked into frames of 45 ms ($N = 300$) with each consecutive frame spaced 15 ms apart ($L = 100$). An 8-pole LPC analysis (autocorrelation method) is performed on each frame of the word (which has presumably been located by an endpoint

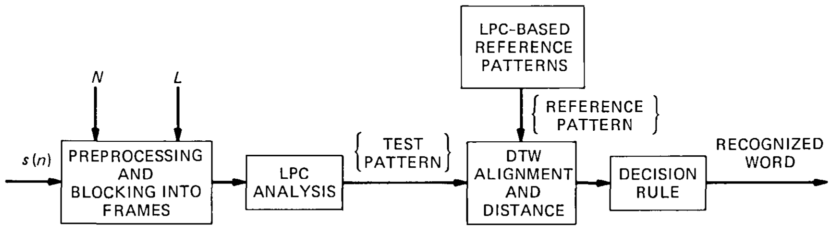

LPC-BASED REFERENCE PATTERNS
N
L
REFERENCE PATTERN
s(n)
PREPROCESSING AND BLOCKING INTO FRAMES
LPC ANALYSIS
TEST PATTERN
DTW ALIGNMENT AND DISTANCE
DECISION RULE
RECOGNIZED WORD

Fig. 2—Block diagram of isolated word recognizer.

detector), creating the test pattern T. This test pattern is compared with each reference pattern using a dynamic time warping (DTW) alignment algorithm, which simultaneously provides a distance score associated with the alignment. The distance scores for all the reference patterns are sent to a decision rule, which provides an estimate as to the spoken word, and possibly an ordered (by distance) set of the best n candidates.

3.2 The robust training procedure

The procedure used to obtain speaker-dependent reference patterns is the robust training procedure of Rabiner and Wilpon.[24] For this method each talker speaks each vocabulary word repetitively (up to six times) until a pair of word tokens are deemed sufficiently similar (based on a DTW distance score). The word reference pattern is created by averaging the two time-aligned versions of the word (the autocorrelation coefficients of each frame are averaged). This procedure yields robust reference patterns since a tight similarity threshold is used to guarantee that the two tokens of the word are free of artifacts by either the talker (e.g., lip smacks, pops, heavy breathing), or from the transmission environment. In this manner it is essentially guaranteed that each of the two tokens being averaged represents a valid pronunciation of the word.

There are two points worth noting about the robust training procedure. The first is that each word is not spoken repetitively until the robust reference pattern can be created. To maximally separate in time the repetitions of each vocabulary word, the talker training the system speaks the entire vocabulary in a random sequence once each pass through the training. The disadvantage of such a procedure is that a considerable amount of storage is required to save the multiple versions of each word that may be required before a robust reference pattern can be obtained. The big advantage of this method is that each word token tends to be an independent pronunciation of the word; hence, word variability is easily and readily measured.

The second point about the robust training concerns the validity of the reference pattern that is obtained. For words with stop releases at the end, e.g., act, back, stop, etc., a speaker will often vary the pronunciation (almost at random). Thus, on some occurrences of these words the speaker will release the stop consonant (leading to a burst at the end of the word) and on other occasions will not release the stop consonant. For such words it should be clear that the robust training procedure cannot adequately represent this dichotomous method of speaking the word and will instead lock onto one of the two variations. For such words a high probability of error is introduced, not by alternate competing words in the vocabulary, as discussed in Section

II, but instead by alternate competing word pronunciations. We see no simple or obvious way of handling this problem.

The robust training procedure trades training time (on the part of the talker) for the ability to obtain robust reference patterns for each word in the vocabulary. For a large vocabulary, notably the 1109-word vocabulary, the average time to train all 1109 words was 5-1/2 hours for each talker! It became imminently clear to the authors that, in practice, one could never consider training speech recognizers for large vocabularies in such a manner. If the need ever arose for word recognition for large vocabularies, an automatic template generation procedure would be required in lieu of the robust training that was used here. Such automatic-template-generation techniques have been used by Mermelstein[25] and by Rosenberg et al.[26] for equivalent size vocabularies using syllable and demisyllable representations of words.

3.3 *Isolated word recognition experiments*

To evaluate the isolated word recognizer on various size vocabularies, a series of word recognition experiments were run. For each of the six talkers, four complete test sets, each consisting of one token each of the entire 1109-word vocabulary, were recorded. The recording took place over four weeks in time, and required about eight hours of recording time for each talker.

The entire data base was used in the first experiment, which consisted of measuring the error rate, E_{1109}, as a function of talker (i), replication (j), and candidate position (n). This experiment provides the absolute performance measure of the word recognizer on the largest vocabulary tested to date.

The next series of experiments basically considered subsets of the 1109-word vocabulary for both training and testing. The Q-word subset of the vocabulary was chosen in several ways to study the influence of means of vocabulary choice on the error rate. The ways in which vocabulary entries were chosen for the Q-word vocabulary included:

(i) Random without replacement—i.e., each of the Q vocabulary words was chosen at random from the 1109-word vocabulary. For each replication of this experiment, the Q words were chosen from the candidates not selected on previous trials. Clearly, a maximum number of trials, $MT = 1109/Q$, is possible with this selection procedure. Since we considered values of Q of 100, 200, 400, and 800, values of MT of 11, 5, 2, and 1 were used, respectively, for the different values of Q.

(ii) Random with replacement—i.e., each of the Q vocabulary words was chosen at random from the 1109-word vocabulary. On subsequent replications a new set of Q words was chosen at random, again from the complete set of 1109 words. For this method of word selection, the same vocabulary word could appear in several replications of the vocabulary. To compare the results of this experiment with those of the one above, the same values of Q and MT were used.

(*iii*) Vocabulary chosen based on best training tokens—i.e., the Q words of the vocabulary, for each talker, were chosen as the Q words (of the 1109) that required the fewest training tokens before the robust reference pattern was obtained. Such words represent the "easiest words to train on," and were expected to be least affected by inherent variability in word pronunciations. Values of Q of 100, 200, 400, and 800 were used.

(*iv*) Vocabulary chosen based on worst training tokens—i.e., the Q words of the vocabulary, for each talker, were chosen as the Q words that required the most training tokens before the robust reference pattern was obtained. Such words represent the "hardest words to train on," and were expected to be most affected by inherent variability in word pronunciations. Values of Q of 100, 200, 400, and 800 were used.

(*v*) Vocabulary with proportional training statistics—i.e., the Q words of the vocabulary, for each talker, were chosen on an equal proportion with their statistics on training. Thus, if a talker had P_2 training words requiring two replications, P_3 training words requiring three replications, etc., then in the test set a total of $(P_j/1109) \cdot Q$ words were chosen at random from the set of words requiring j training replications. In this manner a vocabulary with statistics representative of the training difficulty was obtained. Values of Q of 100, 200, 400, and 800 were used.

(*vi*) Vocabulary with all monosyllabic words. A separate score was obtained using only the $Q = 605$ monosyllabic words in the 1109-word vocabulary.

(*vii*) Vocabulary with all polysyllabic words. A separate score was obtained using only the $Q = 504$ polysyllabic words in the 1109-word vocabulary.

The results of these word recognition experiments are given in the next section.

3.4 Recognition test results

The results of the first experiment, using all 1109 words in the vocabulary, are given in Table II and shown graphically in Figs. 3 and 4. Table II shows values of $E_{1109}(i, j, n)$ for values of i ($i = 1, 2, \cdots, 6$), j ($j = 1, 2, 3, 4$), and n ($n = 1, 2, 3, 4, 5$). Figure 3 shows plots of $E_{1109}(i, j, n)$ versus n for each talker, i, and each replication, j. Figure 4a shows plots of $\bar{E}_{1109}(i, n)$ versus n, where

$$\bar{E}_{1109}(i, n) = \frac{1}{4} \sum_{j=1}^{4} E_{1109}(i, j, n), \tag{10a}$$

Table II—Word error rates as a function of talker (i), replication (k), and word position (n)

	n					
Talker	1	2	3	4	5	k
$i = 1$	12.3	6.3	4.2	3.7	2.9	1
	18.8	9.8	7.2	5.0	4.2	2
	15.6	9.4	6.0	4.4	3.8	3
	12.1	6.4	4.1	2.4	2.1	4
$i = 2$	6.1	2.5	1.4	1.0	0.8	1
	5.7	2.4	1.5	1.4	1.2	2
	6.0	2.1	1.3	1.0	0.9	3
	6.4	3.1	1.9	1.4	1.3	4
$i = 3$	18.4	12.2	10.2	9.2	8.6	1
	19.9	13.1	10.9	9.2	8.7	2
	23.0	15.7	12.3	10.6	10.0	3
	17.8	12.9	10.1	8.9	8.4	4
$i = 4$	40.2	31.2	27.1	24.3	23.4	1
	38.0	29.9	26.1	24.1	22.4	2
	46.7	36.9	32.2	29.8	27.5	3
	48.3	39.9	35.2	32.3	30.1	4
$i = 5$	17.7	10.9	8.7	7.2	6.3	1
	24.8	16.5	12.9	11.0	9.8	2
	20.9	23.5	9.9	7.6	6.9	3
	27.3	17.9	14.2	13.0	11.6	4
$i = 6$	17.0	12.0	9.7	8.3	7.3	1
	17.3	12.4	9.9	8.6	7.7	2
	21.1	15.0	11.8	10.3	9.6	3
	18.6	12.4	9.9	8.8	7.8	4

where $\bar{E}_{1109}(i, n)$ is the error rate averaged over replications; Fig. 4b shows the grand average plot $\hat{E}_{1109}(n)$ versus n, where

$$\hat{E}_{1109}(n) = \frac{1}{6} \sum_{i=1}^{6} \bar{E}_{1109}(i, n) \tag{10b}$$

$$= \frac{1}{24} \sum_{i=1}^{6} \sum_{j=1}^{4} E_{1109}(i, j, n). \tag{10c}$$

Two points in the results are worth noting. It can clearly be seen that within the four replications of a single talker, the error rate scores for a given value of n do not vary a great deal (relative to the absolute error rates). However, across talkers a large amount of variation in error scores is seen for all values of n (see Fig. 4a). Thus, talker 4 has an average error rate of 43.3 percent for $n = 1$, whereas talker 2 has an average error rate of 6.0 percent, a range of over 7 to 1 in error rates.

The grand average (over talkers and replications) error rate curve shows an average error rate of 20.8 percent of the top candidate, and the error rate falls to 9.3 percent for the top five candidates. Although these absolute scores are highly biased by the talker with the high error rate (talker 4), the curves of Fig. 4 show that the error rate for all

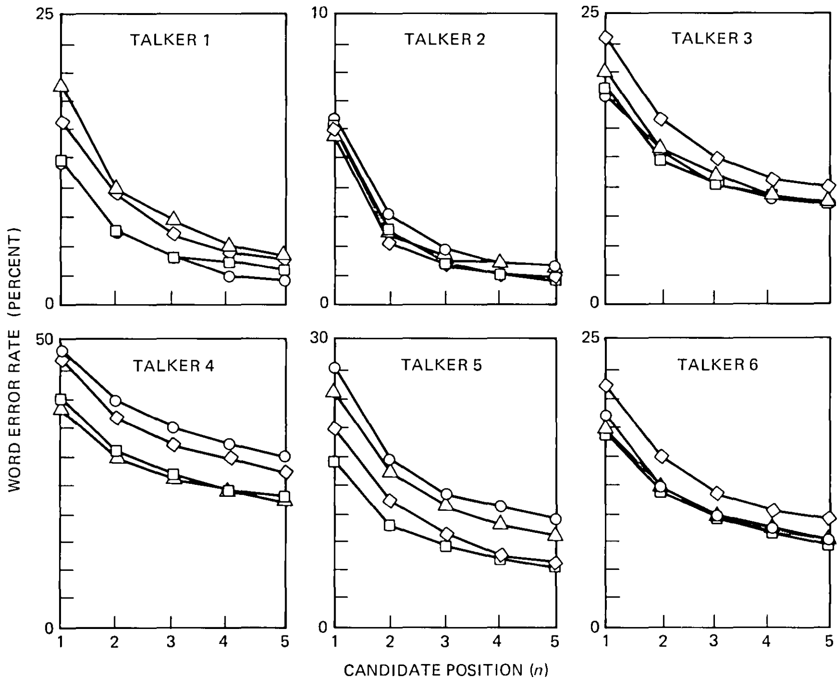


Fig. 3—Word error rate scores for each talker and for each replication as a function of word candidate positions.

talkers displayed similar trends, and hence the grand average curve is representative of the overall behavior of the isolated word recognizer for this vocabulary.

The results of the tests using subsets of the 1109-word vocabulary are given in Table III. For each talker and for each vocabulary partition size Q, this table gives the average error rate (averaged over the four replications) for the top candidate as a function of the subset condition (1 to 7 as described previously). An examination of the data in this table shows the following:

(*i*) Conditions 1 and 2 (random selection without and with replacement) lead to essentially the same error scores on all subsets of the vocabulary for all talkers.

(*ii*) For small vocabulary sizes (Q = 100, 200) selection of vocabulary items based on training statistics leads to very different error rates, depending on the exact set of training statistics used. The error rate scores for condition (*iii*) (best training words) were significantly lower than the error rate scores for condition (*iv*) (worst training words). The error rate scores for condition (*v*) (equal proportions) were essentially comparable to those of conditions (*i*) and (*ii*) and somewhere between those of conditions (*iii*) and (*iv*).

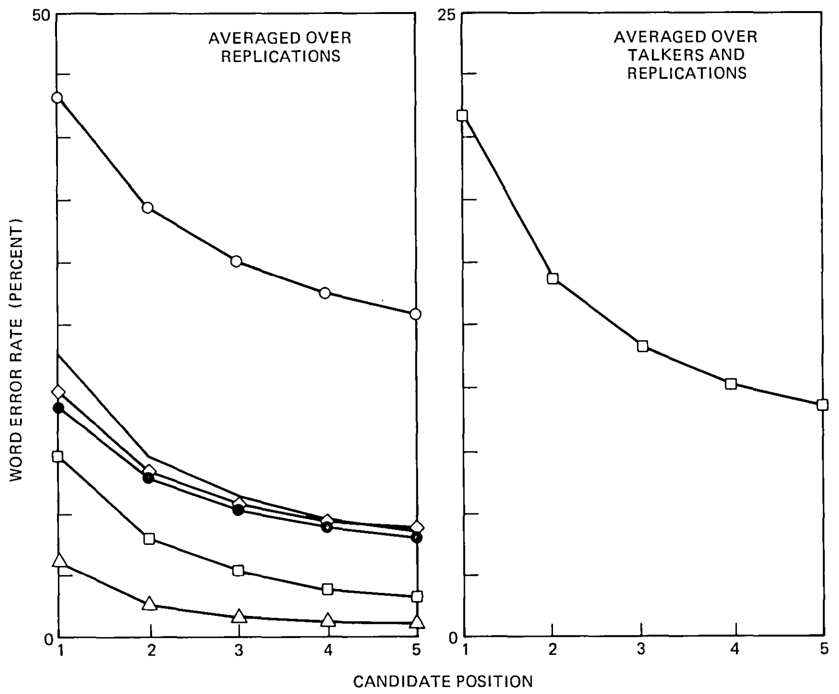


Fig. 4—(a) Average word error rate for each talker as a function of word position. (b) Grand average word error rate as a function of word position.

(*iii*) For the larger vocabulary partitions ($Q = 400, 800$) the effects of choosing vocabulary words based on training statistics on the error rate were small.

(*iv*) The error rates for monosyllabic words alone [condition (*vi*)] were always significantly larger than for any other subset (or even the whole vocabulary) of the vocabulary; similarly, the error rate scores for polysyllabic words alone [condition (*vii*)] were significantly smaller than for any other subset of the vocabulary.

Figure 5 shows a summary plot of the average error rate for each talker as a function of the logarithm of the vocabulary size, and a least squares regression fit to the data points. The data points represent averages of conditions (*i*) and (*ii*) data of Table II. It can be seen that remarkably good fits to the data are obtained, for all talkers, by the least squares regression line. It should be noted that the scales for each talker are different, reflecting the differences in absolute error rates. Similarly, the slopes of the linear fits are different for each talker. In particular the slopes for the individual talkers are 3.1, 1.3, 2.9, 5.7, 4.0, and 3.2, respectively. A slope of α means that for each doubling of vocabulary size, the predicted error rate increases by α percent.

Table III—Average word rates as a function of the partitioning of the vocabulary for each talker

		Q					
Condition		100	200	400	800	605	504
Talker 1	1	3.9	6.4	9.1	13.1		
	2	4.2	5.7	9.4	12.3		
	3	2.5	7.4	7.7	12.2		
	4	5.3	8.0	10.6	12.9		
	5	2.5	6.2	9.2	12.9		
	6					20.8	
	7						6.1
Talker 2	1	1.9	2.6	4.0	6.1		
	2	1.5	2.4	3.2	5.6		
	3	1.5	2.4	2.8	4.2		
	4	3.5	4.6	4.8	5.7		
	5	3.0	2.1	3.8	4.6		
	6					10.0	
	7						2.0
Talker 3	1	10.2	12.9	15.3	18.1		
	2	9.3	10.8	14.7	17.7		
	3	9.0	10.9	12.1	15.2		
	4	25.7	21.6	18.9	18.9		
	5	9.0	11.1	12.7	16.1	29.5	
	6						
	7						7.4
Talker 4	1	23.2	28.0	33.3	40.9		
	2	24.6	29.0	34.6	40.8		
	3	19.8	25.0	29.8	37.5		
	4	31.7	37.2	40.5	43.3		
	5	23.7	24.6	34.2	38.2		
	6					53.4	
	7						28.0
Talker 5	1	8.9	12.0	15.3	20.3		
	2	9.2	11.5	15.5	20.4		
	3	8.0	9.0	13.0	18.6		
	4	18.7	19.5	18.2	21.5		
	5	9.0	11.1	14.2	19.3		
	6					30.9	
	7						11.8
Talker 6	1	7.5	10.0	13.5	17.0		
	2	7.8	10.3	13.4	17.3		
	3	4.7	7.4	10.2	14.3		
	4	15.2	17.1	18.8	18.5		
	5	7.5	8.2	12.6	14.6		
	6					28.2	
	7						6.6

IV. DISCUSSION

The results presented in the previous section demonstrate clearly the effects of vocabulary complexity on error rate for isolated word recognizers. They also show the high degree of variability among talkers in the error rates for almost any size vocabulary.

Perhaps the most startling observation from the data of Fig. 5 is the fact that, for each talker, a doubling in the vocabulary size leads to a

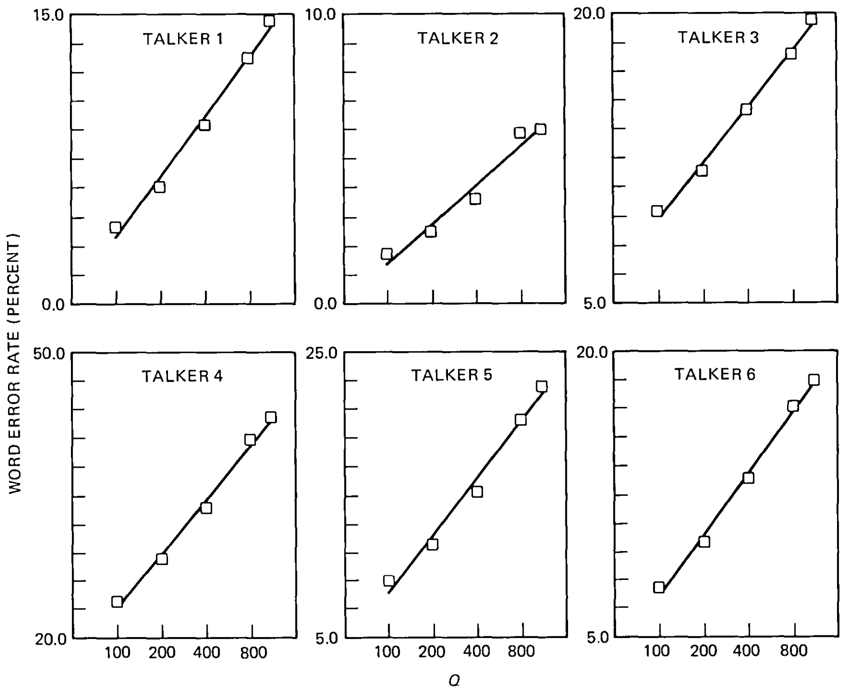


Fig. 5—Average word error rate as a function of vocabulary size for each talker. The straight line is the least squares linear regression fit to the data.

constant (talker-dependent) increase in error rate. This effect has been noted previously by Smith and Erman[27] in their work on word hypothesizing for large vocabulary recognizers. The explanation for this effect is that the error rate is essentially proportional to the density of words in the pattern space [e.g., the factor $(1 - 1/q_i)$ in eq. (6)]. As the number of words in the vocabulary doubles (by random selection), the density increases a constant amount, thereby leading to a constant increase in error rate.

The fact that different talkers have different absolute error rates and different slopes for the same vocabulary sets can be explained by the model of Section II as follows. We postulate that the word similarity threshold, T, of eq. (2) is a talker-dependent threshold in that it is a function of the inherent variability of a talker in repeating a given vocabulary word. For some talkers (e.g., Talker 2) the threshold is set very low and hence very few vocabulary words have q_i values greater than 1. For other talkers (e.g., Talker 4) the threshold is set very high and therefore most vocabulary words have q_i values greater than 1. Thus, the absolute error rate [eq. (6)] will be much higher for talkers with high variability in their word pronunciations than for talkers with low variability in their word pronunciations. Similarly, the increase in error rate for a doubling of vocabulary size is a function (to first order) of the absolute error rate since the density of words in pattern space increases more rapidly for talkers with high word variability than for talkers with low variability.

If the words in the vocabulary are not chosen at random [e.g., conditions (*iii*) to (*vii*) in Section 3.4] then the above analysis is not correct. For example, by choosing words with poor training statistics the average word density is higher than expected, leading to higher word error rates. Similarly, by choosing words with good training statistics, the average word density is lower than expected. Since most words had good training statistics, the effect on the error rate of choosing good training words is generally much smaller than the effect of choosing poor training words. For values of Q approaching 1109 (e.g., $Q = 800$ and $Q = 400$), both effects are smaller since there are only a small number of words (for each talker) whose training statistics were poor.

The average error rates for monosyllables versus polysyllables vividly point out the strong effects of vocabulary complexity. The monosyllable vocabulary of 605 words has a much higher complexity than the total 1109-word vocabulary; hence, it has a much higher error rate for all talkers. Similarly, the 504-word polysyllable vocabulary has a much lower complexity than the 1109-word vocabulary and therefore a much smaller error rate.

V. SUMMARY

In this paper we have presented results of a series of speaker-trained, isolated word recognition tests on a 1109-word vocabulary, and various subsets of the vocabulary. We have shown that although a great deal of variability in error scores was noted across talkers, a fairly good consistency in error scores across replications by the same talker was attained. On the total vocabulary an average (over talkers) error rate of 20.8 percent on the top candidate and 9.3 percent on the top five candidates was obtained. These scores represent the anticipated average performance of the recognizer across different talkers. The best talker achieved a 6.0-percent error rate on the first candidate, whereas the worst talker achieved a 43.3-percent error rate on the first candidate.

By considering various subsets of the 1109-word vocabulary we were able to show that the method of selection of the words within the vocabulary had a strong effect on the word error rate achieved. However, when we used randomly chosen vocabulary subsets all talkers had error rates that increased by a constant percentage for each doubling in the vocabulary size. A simple explanation for this effect was given.

REFERENCES

1. T. B. Martin, "Practical Applications of Voice Input to Machines," Proc. IEEE, *64* (April 1976), pp. 487–501.
2. S. Moshier, "Talker Independent Speech Recognition in Commercial Environments," in Speech Commun. Papers, 97th ASA Meeting, June 1979, pp. 551–3.
3. Interstate Electronics Corp., Voice Data Entry System, unpublished technical descriptions.
4. Centigram Corp., Mike, unpublished technical descriptions.
5. Heuristics Corp., Speechlab, unpublished technical description.
6. W. Lea, ed., *Trends in Speech Recognition*, Englewood Cliffs, NJ: Prentice-Hall, 1980.
7. D. R. Reddy, ed., *Speech Recognition*, New York: Academic, 1974.
8. F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition," IEEE Trans. Acoust., Speech, Signal Processing, *ASSP-23* (February 1975), pp. 67–72.
9. A. E. Rosenberg and F. Itakura, "Evaluation of an Automatic Word Recognition System Over Dialed-up Telephone Lines," J. Acoust. Soc. Amer., suppl. 1, *60* (1976), p. S12.
10. M. R. Sambur and L. R. Rabiner, "A Speaker-Independent Digit-Recognition System," B.S.T.J., *54* (January 1975), pp. 81–102.
11. L. R. Rabiner, "On Creating Reference Templates for Speaker Independent Recognition of Isolated Words," IEEE Trans. Acoust., Speech, Signal Processing, *ASSP-26* (February 1978), pp. 34–42.
12. L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker-Independent Recognition of Isolated Words Using Clustering Techniques," IEEE Trans. Acoust., Speech, Signal Processing, *ASSP-27* (August 1979), pp. 336–49.
13. J. S. Bridle and M. D. Brown, "Connected Word Recognition Using Whole Word Templates," in Proc. Inst. Acoust., Autumn 1979.
14. G. M. White and R. B. Neely, "Speech Recognition Experiments With Linear Prediction, Bandpass Filtering, and Dynamic Programming," IEEE Trans. Acoust., Speech, Signal Processing, *ASSP-24* (April 1976), pp. 183–8.
15. L. R. Rabiner and S. E. Levinson, "Isolated and Connected Word Recognition—Theory and Selected Applications," IEEE Trans. Commun., *COM-29*, No. 5 (May 1981), pp. 621–59.
16. B. Aldefeld, L. R. Rabiner, A. E. Rosenberg, and J. G. Wilpon, "Automated Directory Listing Retrieval System Based on Isolated Word Recognition," Proc. IEEE, *68*, No. 11 (November 1980), pp. 1364–79.
17. A. E. Rosenberg and C. E. Schmidt, "Automatic Recognition of Spoken Spelled Names for Obtaining Directory Listings," B.S.T.J., *58*, No. 8 (October 1979), pp. 1797–1823.
18. P. Vicens, "Aspects of Speech Recognition by Computer," Ph.D dissertation, Stanford University, April 1969.
19. L. R. Rabiner and J. G. Wilpon, "Speaker-Independent Isolated Word Recognition for a Moderate Size (54-Word) Vocabulary," IEEE Trans. Acoustics, Speech, and Signal Processing, *ASSP-27*, No. 6 (December 1979), pp. 583–7.
20. S. E. Levinson and A. E. Rosenberg, "A New System for Continuous Speech Recognition—Preliminary Results," Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (April 1979), pp. 239–43.
21. J. G. Wilpon, L. R. Rabiner, and A. Bergh, "Speaker-Independent Isolated Word Recognition Using a 129-Word Airline Vocabulary," J. Acoust. Soc. Am., *72*, No. 2 (August 1982), pp. 390–6.
22. R. M. Fano, *Transmission of Information, A Statistical Theory of Communications*, Cambridge, MA: MIT Press, Wiley, 1961, pp. 186ff.
23. C. K. Ogden, *Basic English: International Second Language*, New York: Harcourt, Brace and World Inc., 1968.
24. L. R. Rabiner and J. G. Wilpon, "A Simplified Robust Training Procedure for Speaker Trained, Isolated Word Recognition Systems," J. Acoust. Soc. Am., *68*, No. 5 (November 1980), pp. 1271–6.
25. M. J. Hunt, M. Lennig, and P. Mermelstein, "Experiments in Syllable-Based Recognition of Continuous Speech," Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Denver, Colorado (April 1980), pp. 880–3.
26. A. E. Rosenberg, L. R. Rabiner, S. E. Levinson, and J. G. Wilpon, "A Preliminary Study on the Use of Demisyllables in Automatic Speech Recognition," Proc. Int.

Conf. on Acoustics, Speech, and Signal Processing, Atlanta, Georgia (March 1981), pp. 967–70.
27. A. R. Smith and L. D. Erman, "NOAH—A Bottom-Up Word Hypothesizer for Large Vocabulary Speech Understanding Systems," IEEE Trans. Pattern Analysis and Machine Intelligence, *PAMI-3*, No. 1 (January 1981), pp. 41–51.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Multitone Frequency-Hopped MFSK System for Mobile Radio

By UZI TIMOR

(Manuscript received October 12, 1981)

A digital spread spectrum technique employing frequency hopping and multilevel frequency-shift keying has recently been examined for possible application in mobile radiotelephony. Two system parameters, the number of bits per message and the number of tones in the frequency-hopping sequence, are determined by the available bandwidth and the data rate of each user. These parameters in turn determine the tone duration, which strongly influences the vulnerability of the system to transmission distortions. In this paper we describe a generalization of the MFSK scheme that allows users to transmit more than one tone simultaneously. Multitone transmission makes it possible for designers to increase the duration of each tone by increasing the total number of system frequencies. This flexibility slightly increases maximum efficiency and makes the system less vulnerable to multipath delay spread. It could also have implementation advantages.

I. INTRODUCTION

A multiple access modulation technique that uses multilevel frequency shift keying (MFSK) to modulate frequency-hopped spread spectrum carriers has been examined for possible applications in satellite communication[1] and in digital mobile radiotelephony.[2] In every time frame, each user communicates a K-bit message by frequency-shifting a tone sequence that is unique to the user. System efficiency depends strongly on the length of the sequence (L tones per message) and on the size of the tone alphabet. The total available bandwidth and the transmission rate of each user determine an optimum K, L pair; unless the system design parameters are very near this optimum, efficiency is prohibitively low.

As Refs. 1 and 2 report, K, L, and the user bit rate determine the duration of each tone. An important transmission impairment in mobile radio is the multipath delay spread (the difference in delays between the various simultaneous propagation paths). This delay spread makes it difficult to communicate with short tones and, in fact, the 13-μs tone duration prescribed by the optimum K and L derived in Ref. 2 is uncomfortably close to delay spreads observed in practice.

The purpose of this paper is to propose a modification of the modulation scheme that increases design flexibility. By increasing the size of the tone alphabet and allowing each user to transmit more than one tone simultaneously, a designer can increase the duration of each tone, thereby making the system less vulnerable to multipath delay spread without degrading overall efficiency. In fact, there is a slight improvement (10 to 20 percent) in the maximum number of simultaneous users.

For example, with a total (one-way) bandwidth of 20 MHz and users' rate of 32 kb/s, a conventional system can be designed with $K = 8$ bits per message and $L = 19$ tones per message. The tone duration is 13 μs and up to 209 users can operate simultaneously with a bit-error probability less than 10^{-3}. If the bit rate is 31.96 kb/s, $K = 9$, $L = 11$ is possible with a tone duration of 25 μs and 216 simultaneous users. With two-tone operation and $K = 9$, $L = 11$, the tone duration is 26 μs and 225 users can share the bandwidth. Other possibilities are $K = 10$, $L = 6$ (237 users, 52 μs time slots) and $K = 11$, $L = 3$ (224 users, 115 μs time slots).

II. SYSTEM DESCRIPTION[1,2]

2.1 *Principle of operation*

The elementary signals are a set of 2^K tones, each of duration τ seconds. Each link (between a mobile user and the base station) is identified by an address that is a sequence of L K-bit words. A new K-bit message is transmitted every $T = L\tau$ seconds as a sequence of L tones determined by the sum (modulo 2^K) of the K-bit message and the K-bit address words.

The received signal is a composite of the tone sequences of M users. Every τ seconds the receiver performs a spectral analysis of the received signal and decides which of the 2^K frequency cells contain energy. Thus, after T seconds the $2^K xL$ frequency-time received energy matrix (A) is generated. The decoded matrix (A_m) of user m is obtained by subtracting (modulo 2^K) the address words from A (see Fig. 1). The message X_m will appear as a complete row in A_m, at row number X_m. Ambiguous decoding occurs when transmissions by other users combine to form other complete rows in A_m.

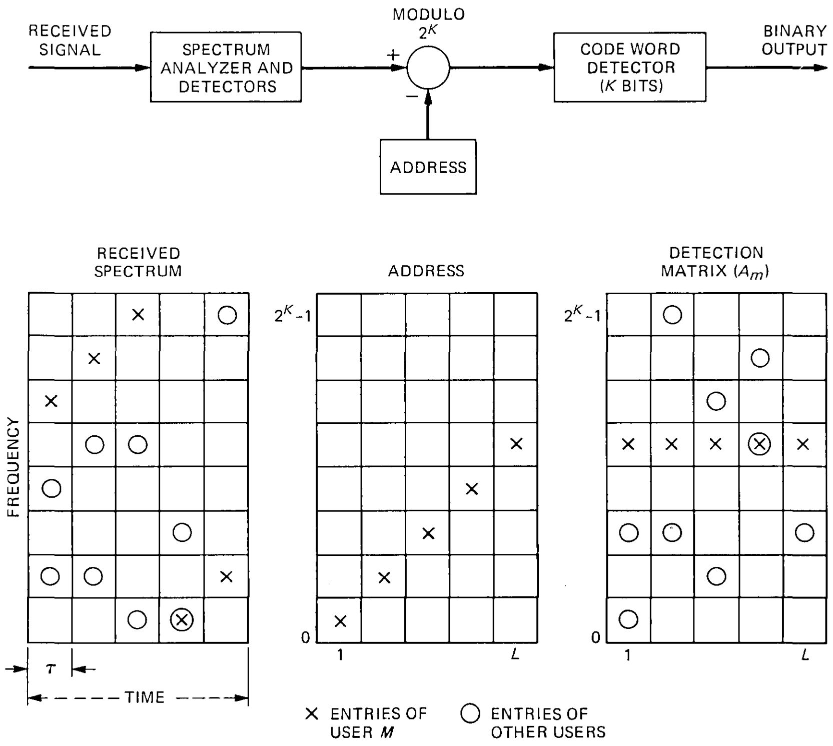


Fig. 1—Receiver block diagram and signal matrices. The matrices show the received spectrum, and the address and decoded matrix of user m, with his entries (X) and those of other users (0).

2.2 System performance

The one-way bandwidth W necessary to support 2^K tones that are mutually orthogonal over $\tau = T/L$ seconds is

$$W = \frac{2^K}{\tau} = \frac{2^K L}{T} \text{ Hz}. \tag{1}$$

The users' data rate is

$$R = \frac{K}{T} \text{ b/s}. \tag{2}$$

For given W, R, and K the length of the sequence L is determined by

$$L = rK2^{-K}, \tag{3}$$

where

$$r \triangleq \frac{W}{R}. \tag{4}$$

Let M be the maximum number of users that can simultaneously share the system at a given error probability. The efficiency η of the system is defined to be the total rate transmitted through the system per unit bandwidth

$$\eta = \frac{MR}{W} = \frac{M}{r}. \tag{5}$$

It has been shown[1] that the bit-error probability owing to mutual interference between M simultaneous users is upperbounded by

$$P_b < 2^{K-2} p^L, \tag{6}$$

where

$$p = 1 - (1 - 2^{-K})^{M-1}. \tag{7}$$

Thus, even without channel impairments the efficiency of the system (or equivalently the number of users that the system can accommodate at a given error probability) is interference limited.

Noisy reception affects the decoded matrix by causing additional entries owing to false alarms and missing entries owing to deletions. Thus, a majority decoder has to be used, i.e., the row with the maximum number of entries is decoded as the message. The performance of the system with Gaussian noise and multipath fading was analyzed[2] and shown to degrade gracefully with increasing noise.

2.3 System design

In designing a spread spectrum system the total available bandwidth and the users' rate (or equivalently their ratio r) constrain the choice of the number of frequencies (2^K) and the sequence length (L) via relation (3). For example, if $r = 626$ the following pairs of (K, L) are possible: (7, 34), (8, 19), (9, 11), (10, 6), and (11, 3).

Substituting (3) into the expression for the bit-error probability [(6) for the noiseless case and a similar expression for the noisy case] yields the optimum pair, i.e., the one that maximizes the number of users that can operate at a given error probability. In the above example, the optimum pair for bit-error probability of 10^{-3} (noiseless case) is $K = 9$ (512 frequencies) and $L = 11$, allowing 216 users.

When a system is being designed, there might be other considerations that would favor changing these parameters. For example, there is a chip synchronization error owing to multipath, and the performance depends on the relative synchronization error (with respect to the chip duration). Decreasing the number of chips (i.e., increasing chip duration) reduces this relative error. Thus, flexibility in the choice of system parameters is desirable. However, a significant deviation from the optimum parameters sharply reduces the efficiency. In the above example, with $K = 10$ ($L = 6$) the number of users is reduced from 216 to 138.

We will show that by allowing each user to transmit two or more frequencies per time slot we obtain this flexibility and thereby improve system performance. We will consider two cases:

(*i*) The frequency band is divided into several subbands and each user transmits one frequency per subband per chip.

(*ii*) Each user transmits several frequencies per chip without any segmentation.

III. MULTITONE SYSTEM WITH SUBDIVIDED FREQUENCY BAND

Consider a conventional scheme with 2^K frequencies and L chips, where $L = 2L_1$ is even. Let us divide the total frequency band into two, each half containing 2^K frequency slots (the number of frequencies is doubled). Suppose each user transmits a sequence of length L by simultaneously transmitting the first L_1 chips on the lower band, and the remaining L_1 chips on the upper band. In the receiver, the two subbands are detected and combined to yield the original $2^K xL$ received matrix (Fig. 2).

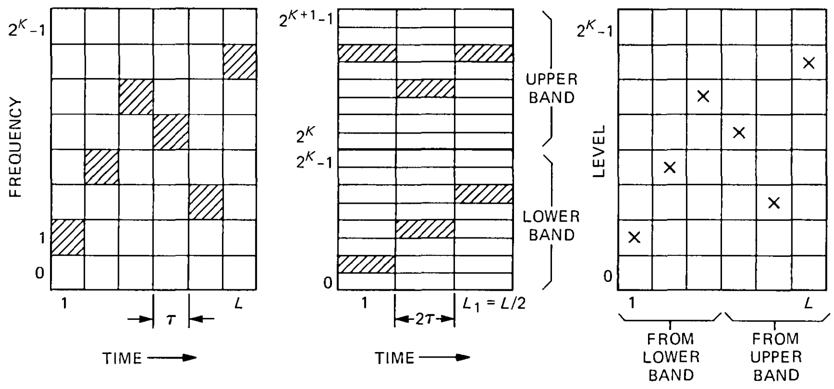


Fig. 2—Transmission of a sequence of L (out of possible 2^K) tones using $L/2$ time slots and 2^{K+1} frequency slots. (a) Generated sequence. (b) Transmitted matrix. (c) Received matrix.

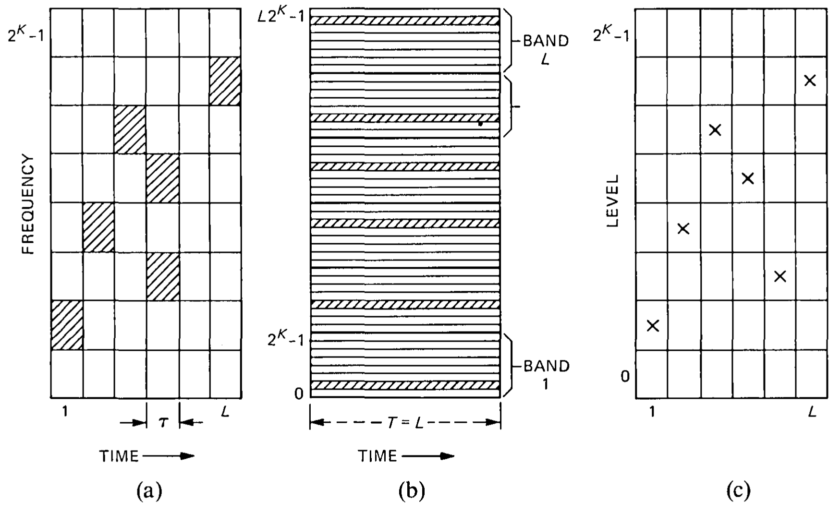


Fig. 3—Simultaneous transmission of a sequence of L (out of possible 2^K) tones using $L2^K$ frequency slots. (a) Generated sequence. (b) Transmitted matrix. (c) Received matrix.

The result is a system with twice the number of frequencies, half the number of chips (the duration of each chip is doubled), and two frequencies per chip.

Similarly, if $L = jL_1$ we can divide the frequency band into j subbands and let each user transmit one frequency per subband. The total number of frequency slots is $j2^K$ and the number of chips is L/j.

In the extreme case we will have one chip (of duration T) and $L2^K$ frequencies divided into L subbands. Each user transmits L frequencies simultaneously, one in each subband. The receiver performs one spectral analysis to determine which of the $L2^K$ frequency cells contain energy. From this, the original 2^KxL received matrix is generated (see Fig. 3).

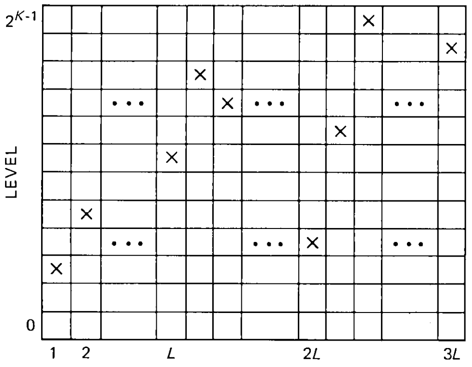
Under the same assumptions, the analysis[1,2] of the conventional scheme for the noiseless and for the noisy and multipath fading channel is valid for the multitone scheme, resulting in a performance that is identical in both schemes.

IV. MULTITONE SYSTEM WITH NONSEGMENTED FREQUENCY BAND

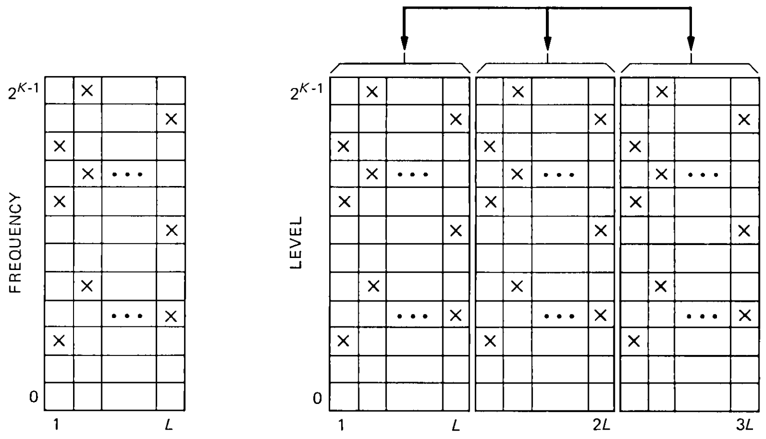
Let each user generate a sequence of length nL, using an address of nL K-bit words. The user then transmits the sequence as follows: At the first time slot, terms 1, $L + 1, \cdots, (n - 1)L + 1$ are sent; at the second time slot, terms 2, $L + 2, \cdots, (n - 1)L + 2$ are sent, and so on.

Thus, each user simultaneously transmits n frequencies at each time slot for a total of nL frequencies (see Fig. 4).

At the receiver, a spectral analysis is performed every τ seconds to generate the $2^K xL$ received matrix. The matrix is then expanded by repeating it n times, i.e., the first column is also entered as the $L + 1$, $2L + 1$, $\cdots$, $(n - 1)L + 1$ columns, and so on, resulting in a $2^K xnL$ matrix (Fig. 4). The expanded matrix is then decoded using the address of length nL.



(a)



(b) (c)

Fig. 4—Transmission of a sequence of length $3L$ over L time slots (chips). Three frequencies are transmitted on each chip. (a) Generated sequence. (b) Transmitted matrix. (c) Received matrix (expanded).

In analyzing the performance of the noiseless case we can follow the same approach as in the conventional scheme. Let M be the number of users and consider the decoded matrix of user m. The probability of having an entry at a particular frequency time slot because of interference from the other $M - 1$ users is

$$p_n = 1 - \left(1 - \frac{n}{2^K}\right)^{M-1}. \tag{8}$$

In addition, at each of the nL columns, we will have, with probability 1, self-interference entries at $(n - 1)$ frequency slots. Let us assume that the self-interference entries in different columns will appear at different rows. [If $(n - 1)nL \ll 2^K - 1$, this can be achieved with a proper address assignment.] Thus, we will have $(n - 1)nL$ rows containing one self-interference entry.

The probability of having a complete row (interference row) at one of those $(n - 1)nL$ rows is p_n^{nL-1}, while for the remaining rows it is p_n^{nL}. Using the union bound, the word-error probability can be upper-bounded by

$$P_W < (n - 1)nLp_n^{nL-1} + [2^K - 1 - (n - 1)nL]p_n^{nL}$$

$$P_W < (2^K - 1)p_n^{nL}(1 + S_n), \tag{9}$$

and the bit-error probability by[1]

$$P_b < 2^{K-2}p_n^{nL}(1 + S_n), \tag{10}$$

where

$$S_n = \frac{(n - 1)nL(1 - p_n)}{(2^K - 1)p_n}. \tag{11}$$

We can combine the two schemes described in Sections III and IV by dividing the total frequency band into j subbands and transmitting n frequencies per subband per chip. In the extreme case we will have a single chip (of duration T) and $L2^K$ frequencies divided into L subbands. Each user transmits n frequencies per subband for a total of nL frequencies transmitted simultaneously, with the performance given by (10).

V. SYSTEM PERFORMANCE

For a given bandwidth to user's rate ratio r, several (K, L) combi-

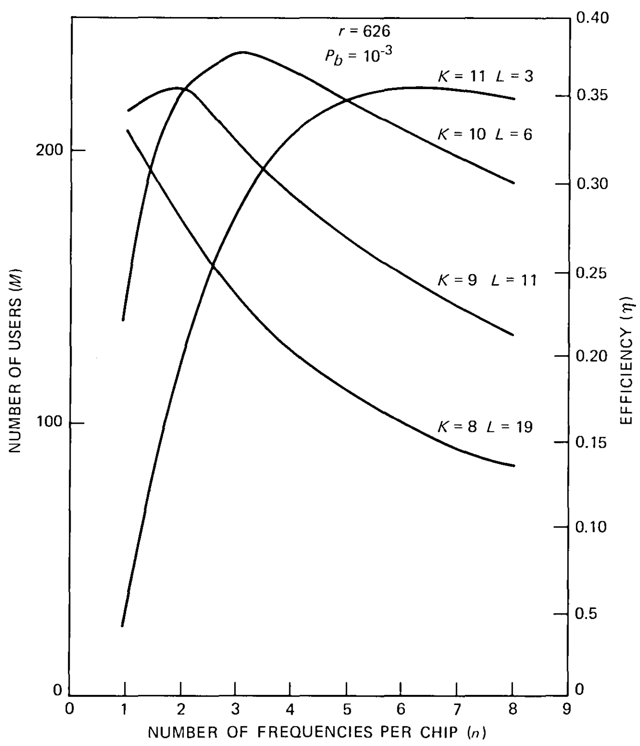


Fig. 5—Transmission of n frequencies per time slot. Shown are the number of users that can simultaneously share the system at bit-error probability of 10^{-3} as a function of n for different (K, L) combinations. The bandwidth to user's rate ratio is $r = 626$.

nations are possible. For each of those, the number of users that can simultaneously share the system at a given error probability depends on the number of frequencies n transmitted per chip.

Figure 5 depicts the number of users as a function of n for $r = 626$, bit-error probability of 10^{-3}, and several (K, L) combinations. Although for $n = 1$ only $K = 8$ or $K = 9$ are reasonable choices, we can now have good performance with $K = 8$, 9, 10, or 11. The optimum is at $K = 10$. The maximum number of users is increased by 10 percent from 216 ($K = 9$, $L = 11$, $n = 1$) to 237 ($K = 10$, $L = 6$, $n = 3$).

The bit-error probability as a function of the number of users for $r = 626$ and several (K, L, n) combinations [$n = 1$ and the optimum n for each (K, L)] is shown in Fig. 6.

Figure 7 depicts the efficiencies of a system with n frequencies per time slot (upper curve) and a conventional system ($n = 1$, lower curve) as a function of r. The best (K, L, n) and (K, L), respectively, were chosen for each r. Also shown are the system parameters K, L, n, and

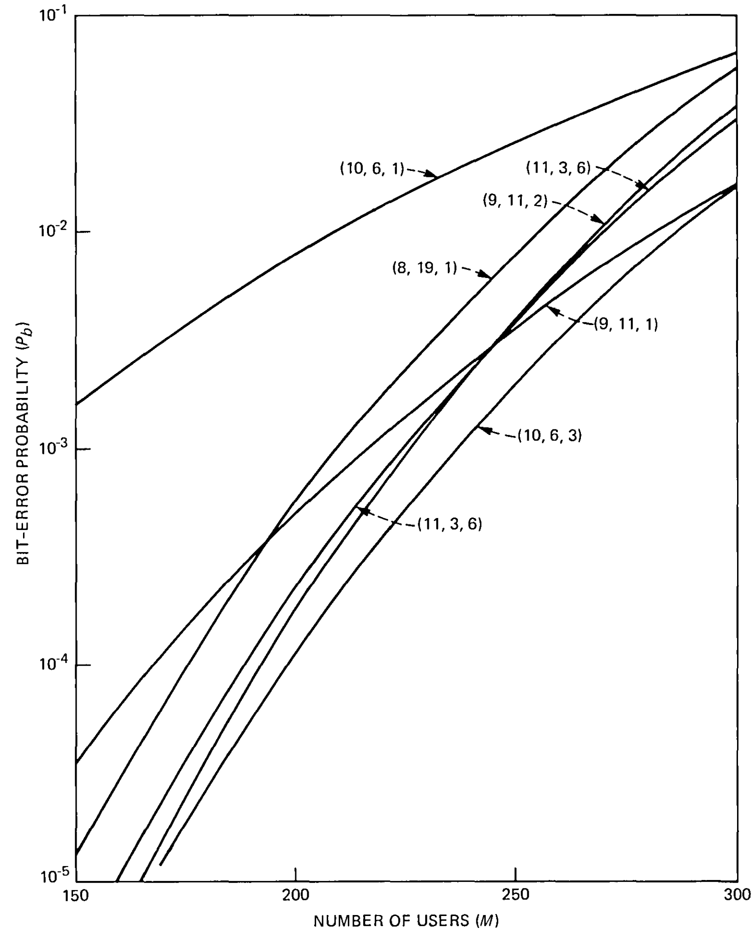


Fig. 6—Transmission of n frequencies per time slot. The bit-error probability as a function of the number of users for various (K, L, n) combinations is shown. The bandwidth to user's rate ratio is $r = 626$.

nL. We can see that for most r there is a 10- to 20-percent improvement in efficiency over the conventional system.

VI. CONCLUSIONS

A modified frequency-hopped multilevel FSK system for mobile radiotelephony that enables users to transmit multitone (instead of single tone) sequences has been presented. This scheme adds more flexibility to the system design by allowing a trade-off between the number of system frequencies and the number of time slots without reducing the efficiency of the system. This flexibility can be helpful in

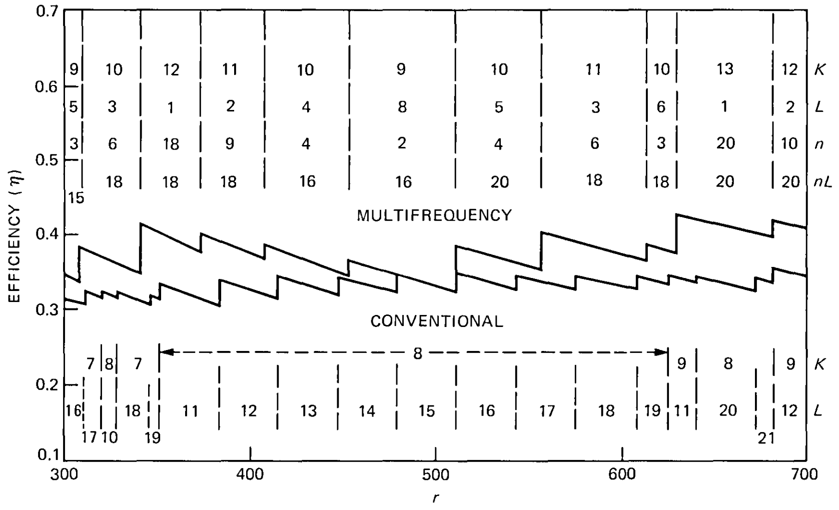


Fig. 7—The efficiencies of a spread spectrum system with n frequencies per time slot, and conventional spread spectrum system ($n = 1$) as a function of r for bit-error probability of 10^{-3}. The best (K, L, n) combination was taken at each r.

making the system less vulnerable to multipath delay spread. It also provides implementation advantages by admitting new combinations of hardware configuration and speed.

REFERENCES

1. A. J. Viterbi, "A Processing Satellite Transponder for Multiple Access by Low-Rate Mobile Users," Digital Satellite Commun. Conf., Montreal, October 23-25, 1978.
2. D. J. Goodman, P. S. Henry, and V. K. Prabhu, "Frequency-Hopped Multilevel FSK for Mobile Radio," B.S.T.J., *59*, No. 7 (September 1980), pp. 1257-75.

Copyright © 1982 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 10, December 1982
Printed in U.S.A.

Design of Recovery Strategies for a Fault-Tolerant No. 4 Electronic Switching System

By R. J. WILLETT

(Manuscript received April 13, 1982)

This paper focuses on the design of recovery strategies that have been developed to give the No. 4 Electronic Switching System (ESS) the necessary high level of fault tolerance. Reliability and availability are the key watchwords to operational integrity in the No. 4 ESS. It is the world's largest toll and tandem switching system, capable of handling 616,000 call attempts per hour, and serving up to 100,000 active terminations. With a downtime objective of less than two hours in 40 years, the No. 4 ESS must be designed to be very fault tolerant viewed from the caller's perspective.

I. INTRODUCTION

The No. 4 Electronic Switching System (ESS) is a high-capacity toll and tandem digital and switching system that is capable of handling 616,000 call attempts per hour and serving up to 100,000 active terminations.

The maintenance system in the No. 4 ESS has the responsibility of responding to error conditions reported by hardware-fault-detection circuits, by memory mutilation detectors, or by some other system-integrity monitor. The strategies invoked for rapid isolation of faulty configurable entities or for correcting memory errors are based on the type of error condition, the state of the system at the time of the failure, and the history of previous recovery actions that may have been attempted.

The recovery strategies employed are fundamentally two-dimensional. One dimension is concerned with the present and the other with the past. Probably the most powerful and unique property of the No. 4 ESS fault recovery system is the latter dimension, that of considering the past, in a process called error analysis. Much more will be said of this process later in this paper.

Although the basic mission of the maintenance recovery system in the No. 4 ESS has remained the same throughout its evolution, the maintenance recovery strategies developed during that evolution have been influenced by many external factors. The availability of new hardware technologies and the development of redesigned and cost-reduced hardware have created the need for making changes in existing strategies. Revisions dictated by field experience and by the introduction of new features have proved emphatically the necessity of being adaptable to change.

This paper discusses the development of fault-tolerant strategies in general, and the state of the art as applied to the No. 4 ESS.

II. FAULT TOLERANCE OBJECTIVE

As we stated earlier, the No. 4 ESS is a large toll and tandem switch capable of handling 616,000 call attempts per hour and serving up to 100,000 active terminations. With a downtime criterion of less than two hours in 40 years, the No. 4 ESS must be, of necessity, a highly reliable system. Recovery actions must be carried out as quickly as possible, and with minimum disturbance to calls active in the system. To put it succinctly, the No. 4 ESS system should be available for processing calls at all times. Any hardware failure or memory error should be detected immediately, before it has a chance to cause call mishandling. From a practical standpoint, this objective cannot be realized under all circumstances, but nonetheless, it is the guiding principle in the design of the No. 4 ESS fault-tolerant structure. Minimizing recovery time and service impact are paramount. Recovery time following the detection of a hardware fault should be measured in terms of 100 milliseconds or less, including the process of detecting, isolating, and restarting the interrupted program. Most faults should be resolved with only one interrupt stimulus. Transient faults (with various manifestations such as transient, intermittent, marginal) may require several interrupts to resolve. Also, multiple fault situations can be experienced, and may take several interrupts for resolution. When such multiple faults can cause limited service impairment, the recovery system must be capable of degrading that service gracefully, i.e., confining the effects to the functional area directly affected.

Memory integrity failures should be corrected by reconstructing valid data from associated information, or if that is not possible, then by reinitializing the memory structure. The former causes less system perturbation, but takes longer; the latter is faster, but can cause more service impact. The choice is usually dictated by the type of structure and by real-time considerations.

III. SYSTEM ARCHITECTURE

Fault tolerance implies system survival under fault conditions. Survival requires alternatives. The architecture employed in the No. 4 ESS is so varied, and the interconnection of components and subsystems is so flexible that several strategies had to be used to achieve the necessary fault tolerance. The design of those recovery strategies takes into consideration system objectives, the hardware technology, system environment, failure rates and failure modes, and the software structure of both the operational and maintenance systems.

3.1 Hardware structure

The basic communications format in the processor and in most of the periphery uses all seems well (ASW) and/or all seems well failure (ASWF) conventions, and single and multibit parity protection over instruction and data transmission. Generally, unique error indicators, such as ASW or ASWF failures, or parity errors are resolvable on one interrupt, unless the fault is transient.

Three basic forms of hardware redundancy are used in the No. 4 ESS architecture to achieve its fault-tolerant objectives: duplication, memory backup, and *n* for *m* sparing. Major common-control elements and critical memory are fully duplicated: the central processor, variable call memory, disk memory, peripheral controllers, and bus access. The switching network is completely duplicated, including both control elements and switching matrix. These duplicated units are operated either in full duplex mode, with parallel operation and possibly matching, or in active/standby mode, with only one unit active and the other ready to assume the active role on demand.

Figure 1 illustrates the redundancy arrangements for the processor subsystem. All three types of redundancy are used: duplication, with and without matching; simplex with backup; and *n* for *m* sparing. Instruction memory, i.e., the program store, and office data memory, are fully backed up on disk, and only one on-line copy of each is provided. In the event of a failure in the on-line copy of office data, one of the duplicate call-memory stores is renamed to replace the failing store, and its contents are pumped from disk. This is a form of *n* for *m* sparing, where *n* is the number of duplicated call-memory units in the system. For program stores, two spare memory units are provided. These, like the office data replacements, can be renamed to replace any program store unit that fails, and the appropriate instruction memory can be pumped from disk.

Figure 2 shows the redundancy arrangements used in the peripheral subsystem. Here again, all types are represented: duplication, with and without matching; simplex with backup; and *n* for *m* sparing. Units that serve a limited number of trunks or special circuits are protected

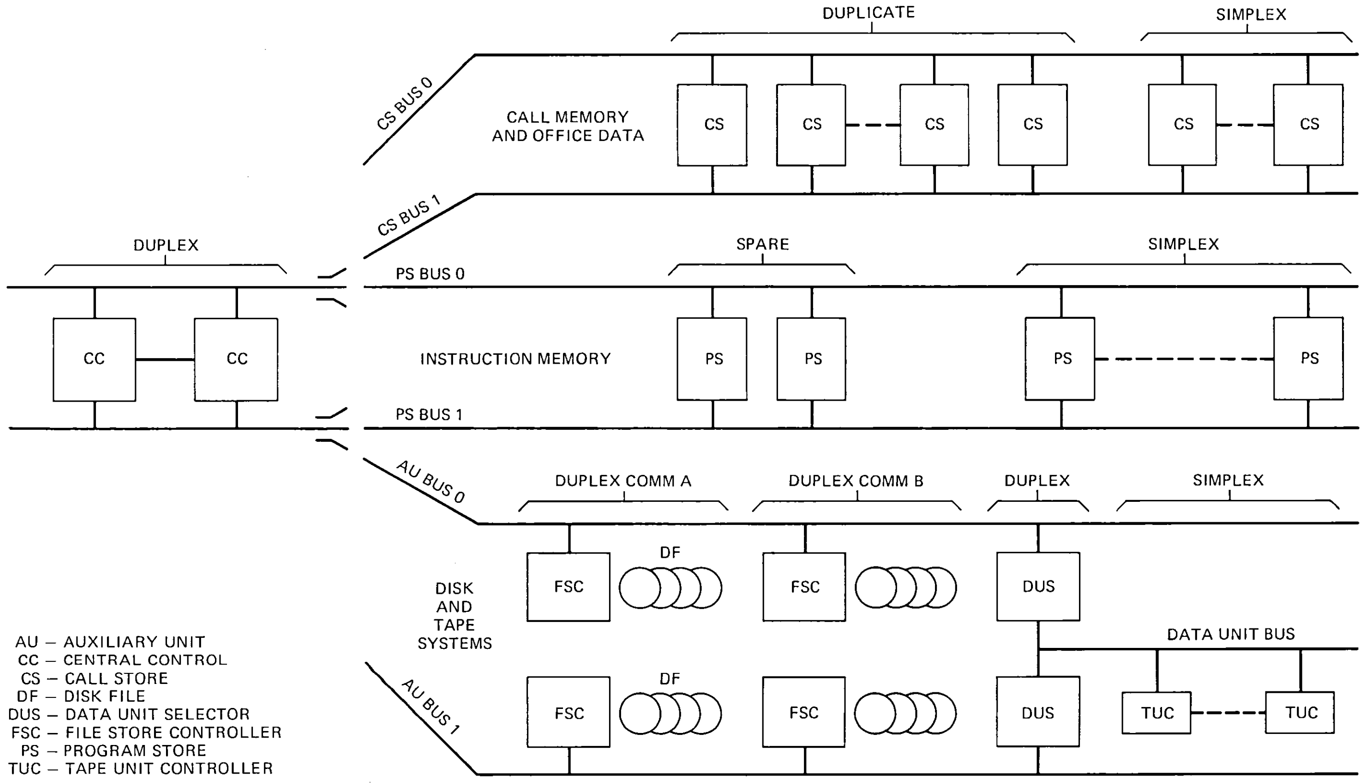


Fig. 1—Processor redundancy arrangements.

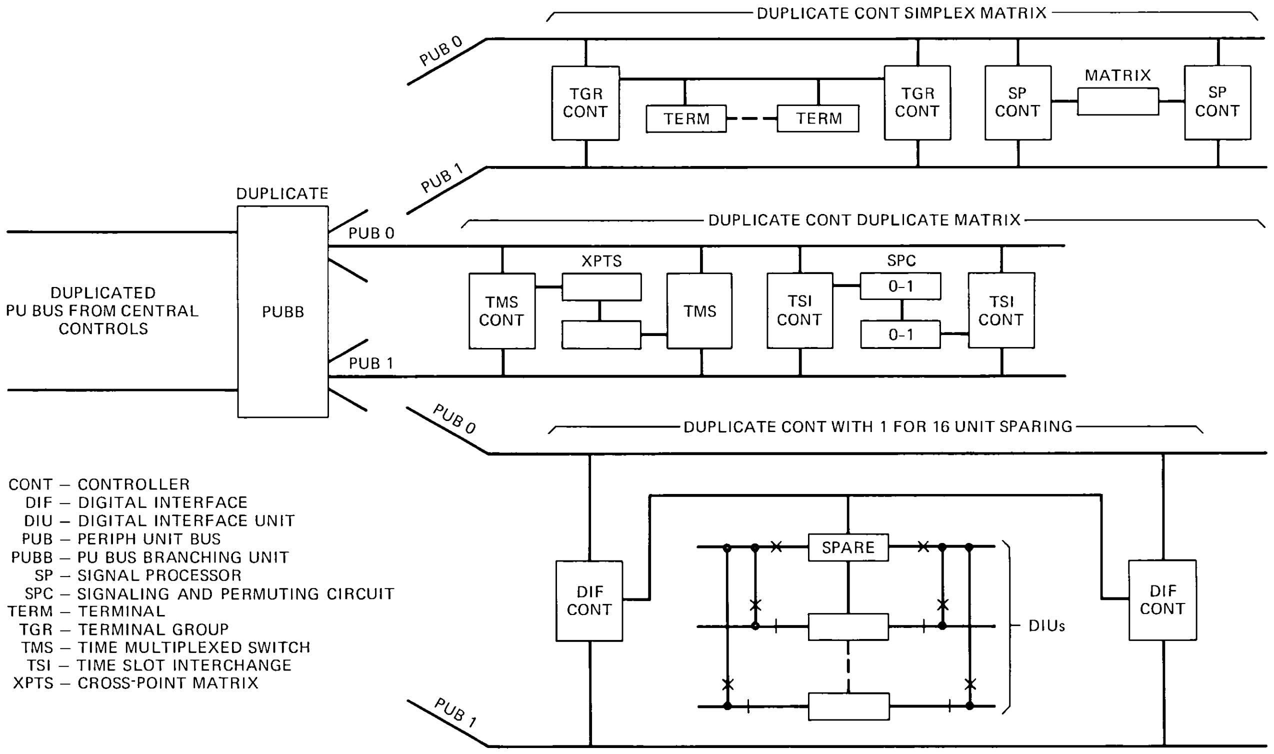

DUPLICATE CONT SIMPLEX MATRIX
PUB 0
TGR CONT
TERM
TERM
TGR CONT
SP CONT
MATRIX
SP CONT
PUB 1
DUPLICATE
PUB 0
DUPLICATE CONT DUPLICATE MATRIX
XPTS
SPC
DUPLICATED PU BUS FROM CENTRAL CONTROLS
PUBB
TMS CONT
TMS
TSI CONT
0-1
0-1
TSI CONT
PUB 1
PUB 0
DUPLICATE CONT WITH 1 FOR 16 UNIT SPARING
CONT – CONTROLLER
DIF – DIGITAL INTERFACE
DIU – DIGITAL INTERFACE UNIT
PUB – PERIPH UNIT BUS
PUBB – PU BUS BRANCHING UNIT
SP – SIGNAL PROCESSOR
SPC – SIGNALING AND PERMUTING CIRCUIT
TERM – TERMINAL
TGR – TERMINAL GROUP
TMS – TIME MULTIPLEXED SWITCH
TSI – TIME SLOT INTERCHANGE
XPTS – CROSS-POINT MATRIX
SPARE
DIF CONT
DIUs
DIF CONT
PUB 1

Fig. 2—Peripheral redundancy arrangements.

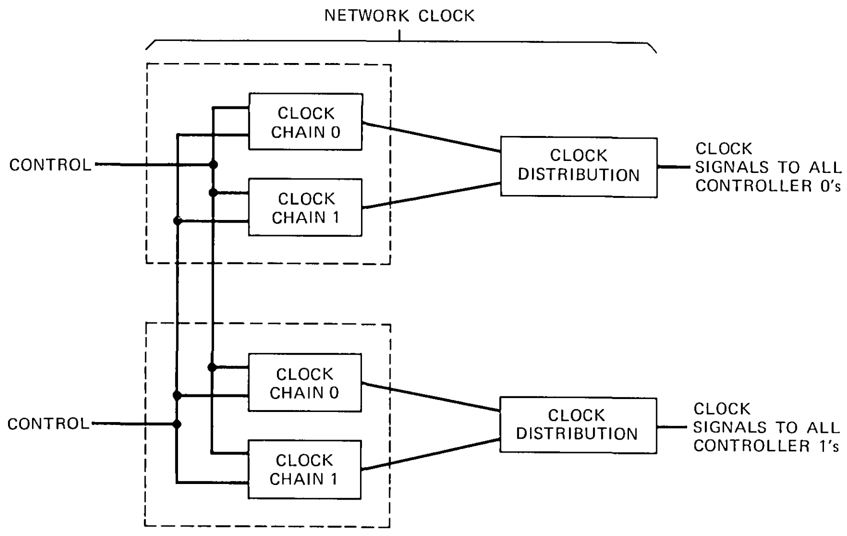


Fig. 3—Network clock (NCLK) redundancy arrangement.

on a limited sparing basis. These generally have one or two spare units that serve as backup for several other units, e.g., one spare digital interface unit (DIU) backs up 16 active units. A spare unit can replace any of the active units by operation of a protection switching mechanism, under control of the fault recovery program system.

The synchronization clock, which is especially crucial to network operation, is quadruplicated. Figure 3 illustrates the special redundancy arrangement used in the network synchronization unit to achieve the high level of fault tolerance required of that unit.

3.2 Software structure

The basic integrity of system memory is protected by audit programs that are structured into mutilation detection and correction modules. The detection modules are constantly run in the background by an audit control program. Additional protection from mutilation is provided by widespread use of defensive checks, e.g., range checks and/or consistency checks in the operational programs. Defensive check failures invoke isolation and corrective responses.

Program integrity assurance is the responsibility of an overseer called the system integrity program. Job scheduling and sequencing are monitored for both frequency and execution times. Program sanity timers are administered. Error conditions are corrected by calling appropriate audits, or by involving various levels of system reinitialization.

IV. ARCHITECTURAL INFLUENCE ON FAULT-TOLERANT DESIGN

The distributed architecture in the peripheral system in the No. 4 ESS poses a unique and often perplexing set of recovery problems. Many peripheral operations are performed autonomously within units that have their own error detection logic. When a fault is autonomously detected, its occurrence is reported to the central processor via a clocked interrogation pulse that triggers a peripheral interrupt. The processor-based recovery control program determines which unit experienced the fault by polling all units using broadcasted interrogation pulses that elicit unique identifying responses from the faulted unit. Once this unit is identified, the appropriate fault recovery actions can be taken. Isolation of these autonomously generated interrupts can be particularly difficult because high levels of subsystem interconnections can cause faults or errors to propagate, and to be reported by other than the unit in trouble. This is especially true in the switching network, where the synchronization clock and switching elements are highly interactive. The network architecture and the associated fault recovery strategies designed to achieve fault tolerance are described in Section V.

Recovery from a fault in the system hardware or from a software error can be viewed conceptually as a three-step operation: detection, isolation, and recovery. Detection is the recognition of the problem. Isolation, in the context used in this paper, is the identification of the failing unit or subsystem. Recovery is the elimination of the fault from the operational system, for example, by reconfiguration or reinitialization. Detection is usually straightforward. Parity errors, ASW-type failures, mismatches, time-outs, and similar triggers stimulate some level of system response such as an immediate interrupt for critical problems, or an interject for less urgent cases. Step two, isolation, is the process of determining the location of the problem. This step is highly dependent on the existence of sufficient indicators to point to the problem area. The particular architecture has a major influence on this isolation process. The third step, recovery, is the heart of the survival process. Here again, architecture is an important consideration in the development of fault-tolerant strategies because it impacts the redundancy plan and the recovery options. Architecture is not the sole consideration, however, because the redundancy plan afforded by the architecture only defines the starting point of the system configuration. The state of the system at the time of an interrupt is equally important, because it tells something of the prior events, and determines what remaining options are available to the recovery strategies. It is useful to look more closely at the architectural influence on fault-tolerant design, particularly as it affects isolation and recovery.

4.1 Effects of architecture on isolation

Isolation of a system problem requires an analysis of the fault indicators used in the detection process, considered in the context of the existing system environment. For a failure in a duplex peripheral unit, the fault isolation program must determine whether one or both peripheral controllers detected the fault, whether both central processors were involved, and whether both access buses were in use. The degree of resolution depends on how much of the available redundancy was on-line at the time of the failure, and, indeed, on the uniqueness of the fault indication. Frequently, an architecture employing real-time matching between duplicate control units will detect a fault without giving a unique indication of which unit half has the fault.

Beyond the question of redundancy, a high degree of interactive coupling exists in the No. 4 ESS, particularly in the switching network. Such tightly coupled architecture poses a real challenge to rapid and effective isolation, particularly if the detection logic does not provide a unique error indication.

Failure modes can affect isolation. This is not in reference to classical gate failures, stuck-at-faults, etc., but failure modes as they may affect accuracy of isolation. For example, a power supply failure, wherein there is high capacitive filtering, can give false indications to the isolation program because of the time constant in the power supply filter. This can occur because some logic gates fail before others as power drops from its proper level towards zero. As another example, an autonomous peripheral unit may trigger an interrupt in the central processor, but then, by the nature of the fault, be unable to respond to the interrogation pulse sent to identify the failing unit. As a further example, there have been experiences where a control unit, whether by timing idiosyncrasies or by the nature of the fault, entered a state where it would not respond to any test commands until the unit was forced into an initial state by physically cycling power on it. These are but a few examples of how failure modes can affect isolation.

Isolation of a software problem is also affected by architecture. Autonomous peripheral units control many operational and maintenance processes, and pass data to the central processor via report buffers. This architecture, using buffers, requires the program monitor system to screen out invalid reports, out of sequence reports, and to correlate the data to the structures that could be implicated. The importance of this issue has become intensified with the addition of peripheral microprocessors and adjunct processors, with their loosely coupled architecture and further delayed buffering of information.

4.2 Effects of architecture on recovery

The issue of recovery, interestingly, is more difficult to resolve than that of isolation because service objectives and architecture play a strong role in determining whether the resolution can be carried out. Isolation is the process of determining where the fault lies, whereas recovery is the process of removing the faulted unit, if possible. The *if possible* is the key issue in recovery. If the active half of a duplicated unit is the only available half, then removing that half would be tantamount to degrading service. Such action may have to be taken eventually, but service impact, accuracy of isolation, etc., must be considered first.

V. DESIGN OF FAULT-TOLERANT STRATEGIES

The foregoing discussion of the influence of architecture on the isolation and recovery process has only touched upon the real question of how to design fault-tolerant strategies for the No. 4 ESS, or, for that matter, any large real-time, high-availability system.

A basic premise of ESS maintenance is the single-fault assumption. Under normal circumstances the mean time to failure of any part of the hardware is much greater than the time to repair that failure; therefore, no part of the system should experience simultaneous multiple faults that would impair service. From this it follows that fault recovery should be a simple one-dimensional process.

Then why is it not always so? The architectural redundancy plan came from the basic premise. Critical units, including most common control elements, are fully duplicated. Memory is either duplicated directly or backed up in some other medium, e.g., disk and random access stores. Other units, particularly units that interface with trunk circuits, are engineered with one or more switchable spares, determined by reliability considerations and service objectives. Normally, all redundant elements are configured for maximum availability. A classical stuck-at fault occurring within such an environment will normally be detected and uniquely isolated from a single interrupt stimulus.

5.1 Basic duplex recovery strategies

When a duplex unit causes a system interrupt, recovery is effected by isolating and removing the faulted unit from service, and activating the redundant half, consistent with the remaining system environment. This same approach applies to units with backup or spare redundancies. If the redundant unit is available, it will be pressed into service when the active unit fails. The recovery action is concluded by a request for a full diagnostic test of the faulted unit to determine the location of a replaceable module. Once repaired, the unit will again be diagnosed, and returned to duplexed operation, if it passes the diagnostic tests.

The preceding scenario describes the simple process of fault recovery as it occurs most of the time. Why does it not always happen this way? It is simply because of the vagaries of the system. The single-fault assumption is true most of the time. But, it does not always take a fault to put the system into a simplex mode of operation. Much of the hardware contains maintenance logic used to detect faults. The operational logic is exercised by executing operational commands, but what exercises the maintenance logic? Generally, much of the fault detection logic is unchecked during execution of normal commands. To ensure the integrity of the fault detection capabilities, most of the logic is periodically tested during low traffic periods. During such tests, the unit being diagnosed is usually removed from service, thus foregoing full-duplex operation for the duration of the routine tests.

The frequency of the routine testing takes into account the length of the test, and the residual reliability of simplex operation to the system. Many of the units may be routinely tested on a daily basis, some every few days to a week. While a unit is in simplex operation, recovery from a fault in the active control half requires a different, more subtle, strategy than the simple duplex strategy described earlier.

5.2 Basic simplex recovery strategies

When a unit half is removed from service and a diagnostic test of that unit scheduled, the recovery program will, where possible, try to put the suspect unit into a special out of service mode. In this mode the recovery program's internal memory and sequencers will track the active unit in case the recovery operation had implicated the wrong control unit. Such a mode is referred to as a listen-only mode (LOM). If, before the start of the diagnostic tests, the active unit should interrupt, the LOM control unit can be put back into service immediately, without the need for updating it, and the mate control unit can be removed. Once the diagnostic tests have started, of course, the out of service control unit becomes out of date and cannot be put back into service without updating or reinitializing it.

Consider the first of these two cases just cited. If the fault recovery program removes the wrong control unit on the first stimulus from a mismatch error, the fault in the active unit would likely cause an immediate second interrupt, while the out of service unit was still LOM. The second recovery action would restore the previously removed control unit, remove the active unit, and request diagnostic tests. In the second case cited, with the out of service control unit out of date, a fault in the active unit can have serious service repercussions. Reinitializing the out of service control unit, particularly its memory, will cause the loss of all calls that may be in progress within that unit.

Considering that a unit may be serving as many as 4,000 trunks, or possibly even a quarter of the calls active in the switching network, reinitialization of a control unit demands serious deliberation. To protect against undue service impact, the basic simplex strategy turns to considering the impact of the present interrupt. Could this be due to a transient fault? How long has it been since the last interrupt? Presumably, if the rate of interrupts is sufficiently low, the system could tolerate an occasional interrupt, and a possible mishandled call, better than it could a major service disruption. To estimate the service impact of the present interrupt, the fault recovery program uses a Leaky Bucket Counter (LBC) to judge the rate at which the interrupts are occurring. For each interrupt that is experienced, the LBC is incremented or decremented by an amount that is determined by the elapsed time since the last interrupt. The higher the interrupt rate, the faster the LBC will reach a predetermined threshold, and reinitialization of the out of service control unit will take place. If the interrupt rate is low, or of short bursts, the LBC may never reach threshold, and the system will tolerate the perturbations.

When a hard or persistent fault causes the LBC to reach its threshold, a reinitialization, or zero-start as it is called, will be requested on the out of service control unit. This action will cause all calls in the unit to be lost. Additional memory reinitialization outside the unit may also be required, as in the case of the switching network with its network maps, and with trunk interface units with the trunk state memory.

The basic simplex strategies provide a high measure of fault tolerance for the No. 4 ESS for hard faults, and use the redundancy in the system for maximum service availability. For software errors, transients, and several other situations that may or may not be unique to the No. 4 ESS, these strategies are, however, found wanting or inappropriate. Further, there are other major considerations that affect survivability: changing fault data, shifts in strategy, false conclusions, etc. These, and other important dimensions, will be considered later in this paper. First, however, the software error and transient strategies will be discussed.

5.3 Recovery from software errors

System problems can be caused by at least two types of software errors: program anomalies (bugs), and bad data. There is a better chance of coping with bad data than with program bugs, but there are recovery strategies in the No. 4 ESS to deal with both.

When a call-handling program tries to address a non-existent unit within the No. 4 ESS, an ASW interrupt will occur because there is no unit to return the expected affirmative response. Presumably, the program that issued the command is working with bad data. The ASW failure triggers the fault recovery process, but in this case, the trouble is not a fault but a program error. No hardware recovery action is therefore required. The appropriate reaction is to correct the data being used by the call-handling program, or to kill the call, or both if necessary. The recovery strategy will call on the memory audit system to run one or more relevant audit programs to detect and correct memory inconsistencies that may exist. The recovery program will avoid returning control to the program that issued the order, but rather will return to a safer control point. Despite that precaution, it is possible that the same program could regain control at a later time and cause the system to take additional interrupts before the audits can completely correct the problem. To allow the audits a chance to correct the problem, the recovery strategy may decide to inhibit, or prevent, further interrupts of that type. This process is called pesting interrupts.

Not all interrupts caused by bad data occur as a result of invalid commands to non-existent units, causing what is commonly called an out-of-range (OOR) failure. Many interrupts that are in range, i.e., from valid units, can be isolated to software causes. Again, appropriate audits are called by the recovery strategy. One difference, however, is that local- (frame) level pesting of interrupts can be used, rather than the system-level pesting used in the OOR case.

Running pested, or with interrupts inhibited, is risky, at best, because some fault detection capability is disabled. Such risks are justified, however, under the single fault (or error, in this case) assumption.

One can carry the software issue a step further in developing a recovery strategy to meet all cases. There often exists a possibility that the resolution of a software error is, in reality, not true, and that the real problem is indeed a hardware fault. The accuracy of the resolution is not infallible. Using the software recovery strategy described does not take into account the possibility of a hardware fault, and therefore would not resolve the issue if that were true. In this case, at some point in the recovery sequence, one of the involved control units would be removed from service and diagnosed. If neither control unit is found faulty, the final action of pesting the frame would be invoked.

When an interrupt is caused by a program bug and not by bad data, the previous strategies of calling audits and/or removing hardware may not resolve the problem. The final action of pesting the system will, at least, allow continued system operation, albeit in a possibly degraded or more vulnerable environment. In many instances where a program bug has caused problems, running pested has given maintenance personnel enough time to analyze the fault recovery data and make operational adjustments. These adjustments could be in the form of immediate program corrections, procedural changes, or temporary program changes to pin-out or bypass the offending program.

5.4 Basic transient strategies

It is generally easier to deduce that a fault is transient than it is to isolate it. The results of retrying a failing command can usually establish whether a given fault condition is hard or transient. The problem is that when the fault appears to be transient, there is some doubt that the proper suspect has been determined, or if it is the right suspect, the diagnostic tests will find nothing wrong. This leaves the strong possibility that the fault can plague the system for some period of time.

There are two primary strategies for coping with transient faults in the No. 4 ESS, one to cover the case where the isolation program is reasonably certain of the suspect unit, and another to qualify the decision with a measure of uncertainty. In both cases, on the first interrupt, the primary action is to record the error and clear the error conditions in the suspect unit. If a second interrupt occurs, the suspect will be removed from service, provided the duplicate mate, or backup unit, is available. The suspect will be put into the LOM state to ensure that it will remain up to date in the event of a wrong decision. Cyclic diagnostic tests will be scheduled on the suspect unit, expecting that a transient fault might not be caught by a single set of tests. For example, diagnostic tests might be repeated three times, to improve the probability of isolating the fault.

The two transient recovery strategies diverge at the point of the third interrupt. If subsequent interrupts still occur beyond the second interrupt, and the isolation program indicates uncertainty, the recovery strategy will change the suspect, and remove and call for a cyclic diagnosis of the mate control unit.

Note that it is the diagnostic test results that drive the escalation of the recovery strategy. Each time a transient fault escapes detection by the diagnostic tests, the unit is restored to service. The strategy has a built-in control that will limit the number of times the cyclic diagnostics can be scheduled. Beyond that point a suspect unit will be removed from service without further testing. It is left to the skill of the maintenance personnel, using special test facilities, to trace the problem further.

The transient recovery strategies covered above are used when the suspect and its mate are both on-line and available at the time the interrupt occurs. If a transient fault causes an interrupt while a unit is simplex, the basic simplex recovery strategies described earlier apply. When redundancy is not available, the recovery strategy is insensitive to whether the fault is hard or transient.

5.5 Special problems

Several other fault-tolerant strategies have been developed for the

No. 4 ESS to cover situations that are beyond the scope of the more general strategies described above. Buffer overflow, network interface problems, internal network failures, and many other cases require special recovery strategies.

5.5.1 Buffer overflow

Autonomous processors operating asynchronously communicate with the central processor via report buffers. Buffer sizes are engineered to provide enough capacity to absorb spurts in activities and occasional delays in retrieving reports from the buffers. Critical buffers are protected by overflow detection logic that triggers an interrupt, or more likely, a lower priority interject, and calls a fault recovery sequence into action. Possible causes of buffer overflow could be related to program integrity, i.e., the report handler not being scheduled, unusual traffic overload or congestion, or even an undetected hardware problem.

The primary recovery strategy covers two probable causes. It schedules the report dispenser program to unload the hyperactive buffer, and also calls for an audit of the task sequencer control tables. To prevent any further overflow triggers from firing before the recovery action is completed, the buffer overflow detection logic in that unit is pested, i.e., inhibited, for some short period of time. If overflow conditions continue despite the recovery attempts cited, the hardware becomes suspect, and appropriate actions to remove a control unit are attempted, consistent with the strategies described earlier.

5.5.2 Network interface problems

There exists a number of units in the No. 4 ESS that are situated between the switching network and the trunk circuits and that can be classified generically as the network interface units (NIUs). Functionally, these units are responsible for separating the signaling and voice or data information, and converting the voice/data from the incoming or outgoing transmission format (analog or digital) to the pulse code modulation (PCM) format switched by the No. 4 ESS network. The NIUs interface with the switching network via coax cables, each carrying 120 trunks time-multiplexed onto 128 time slots. Several NIUs are protected by one spare unit that can be protection-switched into service by the recovery program. The number of units served by each spare varies with the type of unit in question, but it ranges from one spare for six units to one for 16.

While the NIU recovery problems may be unique to the No. 4 ESS, or possibly other ESS machines, the fault-tolerant strategies developed for this purpose may have relevance for other $1/m$ or n/m redundancy arrangements.

A fault occurring in a NIU can be detected either by the NIU control unit or by the switching network, depending on the location of the fault. Regardless of how the fault is detected, the basic recovery concepts for hard or transient faults apply. The pointed difference in strategy comes about when the recovery decision is to remove the faulted unit. If the spare NIU is available, it will be switched into service, much as is the mate of a duplicated control unit. If the spare NIU is presently serving in place of another NIU, or if the spare is out of service, then the active NIU cannot be removed without degrading service for the 120 trunks served by that NIU. In that event, on the first interrupt, only clean-up action is taken. On later failures, the fault recovery strategy tries to optimize the use of the spare NIU, and if possible move the spare to replace the present suspect NIU. If that cannot be done, then the suspect NIU is removed from service and the 120 trunks are correspondingly taken out of service and marked "maintenance busy."

An interesting extension of the recovery strategies for NIU failures is invoked when multiple NIU errors are detected simultaneously. On multiple errors, the recovery strategy escalates to implicate one or even both of the NIU common control units, in place of, or in addition to, the NIUs themselves.

5.5.3 Internal network failures

Perhaps the most sophisticated fault-tolerant strategy developed for the No. 4 ESS involves the isolation and recovery from internal errors in the Time Division Network (TDN). This switching network includes a four-stage space switch with time-switch matrices as initial and final switching stages. The PCM voice/data bit streams are time-multiplexed onto 128 time slots, and are protected by simple parity and a leading-one protocol.

The components of the TDN are two units that provide time-space and space-time switching stages, one for transmitting and one for receiving, located on either side of a unit that provides a two-stage space-switched matrix. The network topology is shown in Fig. 4. The TDN architecture is a multi-dimensional, tightly coupled complex. Both the time and space switching units are fully duplicated, including the switching matrix, and are normally operated in a full-duplex, matching mode. Either half of the transmitting or receiving switch units can communicate with either half of the space-switched unit.

A special problem posed to the No. 4 ESS fault recovery system is the problem of resolving the class of internal TDN errors called transmit/receive parity failures (TRPFs). A TRPF is a failure in one of the talking paths, i.e., time slots, caused by a fault somewhere in the TDN matrix elements. On a given TRPF interrupt, six possible sus-

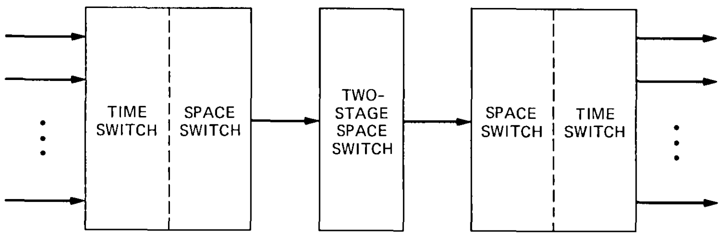


Fig. 4—Time division network topology.

pects exist (one space-switched and two time-space-switched units, duplicated).

In a well-behaved, single-fault situation, i.e., a readily isolated hard fault, the isolation program will determine which elements of the TDN were involved in the failed talking path and set up all possible combinations of links between the various duplicate matrix elements. By a process of elimination, the isolation program can determine which combinations experience an error and which do not. If all combinations are available, and a single fault exists, a unique isolation can be accomplished on one interrupt, and recovery effected by removing the faulty control and matrix half. Again, as with previous strategies, diagnostic tests are scheduled to facilitate repair.

Three conditions can short-circuit this network recovery strategy: the fault is transient (more accurately, marginal), the fault is multiple, or the TDN was not fully duplex at the time of the interrupt. Any one of these conditions prevents unique isolation and accurate resolution on a given interrupt. A special recovery strategy has been developed to cope with this problem.

On each unresolved TRPF interrupt, the recovery program updates special leaky bucket counters for the three (and possibly six) units/matrices involved in the failed talking path. Such records are maintained on each unit in the TDN. The first unit to exceed a predetermined leaky bucket threshold is deemed suspect, and its control unit is removed and diagnosed. The assumption is that calls will be evenly distributed throughout the network, and if any one unit is involved in TRPF failures above all others, then the weight of suspicion falls on that unit. Diagnostic tests will confirm or deny that suspicion.

5.6 Practical considerations on total fault-tolerant design

All the fault-tolerant strategies previously considered addressed specific problems of isolation and recovery. There is a whole area of concern in the design of a total fault-tolerant system beyond that already discussed.

5.6.1 Inconsistencies in fault resolution

Questions arise about the continuing strategy to use if the fault signature or resolution changes as various recovery attempts are made. How long should the failure history actively be maintained on a unit, particularly after a fault apparently has been repaired?

In the deductive process of isolating a fault in the No. 4 ESS peripheral system a tree-branching structure of decisions is used based on the initial error source data and the results of tests, with the resolution being declared a point of maximum definition (PMD). Theoretically, the PMD data developed from these tests yields a unique identification of the faulted unit, the class of fault (e.g., hard-unique/non-unique, transient, software), and whether the fault was resolved or not resolved. It then gives a recommendation of the strategy to be followed for recovery. The recommendation is based on the current interrupt and current circumstances, with no prior events taken into consideration. The isolation (PMD) data and recommendation are passed to an error analysis program that has access to history records of previous interrupts and recovery actions. If the current interrupt is the first such occurrence, the error analysis program will allow the recommended recovery strategy to be carried out, insofar as it is consistent with service criteria. If, however, a previous history of interrupts exists for the suspect unit, it is important to determine if the current recommended strategy is the same as that followed on previous interrupts. If it is not, which strategy is it appropriate to follow?

There are many reasons why the resolution, and consequently the recommended recovery strategy, can change from one interrupt to the next. The fault could be data sensitive, the error indications could be sensitive to the state of the unit, or the problem could even be an actual or apparent multiple fault situation. In any event, there are cases where the previous recovery strategy should be continued, and some where the current recommendation is more appropriate. Also, sometimes the current PMD data could preclude a continuation of the previous strategy because of data and/or environment incompatibilities.

A common situation would be for a transient fault to later develop into a hard fault. Different strategies for recovery are followed for each case, as described earlier. In this instance, it is reasonable to change from the transient strategy to a hard-fault recovery sequence because the latter addresses a specific rather than a nonspecific causal relationship. If the reverse were true, that is, if the resolution of a particular interrupt indicated that the fault was transient when all previous experiences were seen to be hard, then it would probably be inappropriate to abandon the previous strategy. Many conditions could alter this position, however. Is it the same fault in each case? Had repair been attempted prior to the latest interrupt? How much time had elapsed since the previous interrupt?

The point of these questions is to determine if two successive interrupts are caused by the same fault, when the resolution in each case might suggest otherwise. Because of these natural state transitions (transient/hard), and for countless other causes, it is necessary to recognize when different recovery strategies are indicated on successive interrupts and to determine which of the two strategies to follow.

Before the process used in the No. 4 ESS for escaping between recovery strategies is described, an explanation of the process for recognizing the existence of prior associated events and for comparing the past and current recovery strategies will be given.

5.6.2 Error Analysis

The isolation of a fault or other cause of an interrupt is under the control of the Fault Recovery (FR) programs. As explained earlier, the FR programs isolate the fault or error using a deductive process consisting of retries or specific tests under various configurations. The results of these tests are passed to the failure error analysis (FERA) programs as PMD (point of maximum definition) data, detailing the identity of the suspect unit or cause, the class of fault, and indeed, whether the problem was resolved. The data further includes a recommendation of a recovery strategy to follow. Keep in mind this PMD data is based strictly on the current problem. The FERA programs represent an extension of the recovery process. As each interrupt is processed, a record is retained of the PMD data and the final action taken. When an FR program isolates the cause of a current interrupt, FERA searches all active history records for data implicating the same unit. Once found, the past and present recovery strategies are compared with entries in a strategy escape table. If the strategies are the same, or if the previous strategy is still more appropriate, escape will be denied, and FERA will continue to follow the existing recovery strategy. If the new strategy recommended by the FR isolation program is appropriate to the fault class and status of the suspect, the escape data will allow FERA to abandon the previous strategy and start the new recovery sequence.

The question of escaping from one strategy to another is not solely an issue of changing resolutions, but also can be an issue of changes in the unit state. For example, an FR resolution might call for the removal of a suspect unit, and under a duplex unit availability, FERA could concur. Under a simplex unit configuration, however, the same recommendation, if followed, would cause service degradation. The escape table data will cause FERA to deny the initiation, or the continuance, of a duplex strategy, when a unit is no longer configured in duplex. In such a case, an alternative strategy is not necessarily provided in the PMD data. FERA must make that determination from strategy selection tables provided for that purpose.

The issue of whether successive interrupts from the same unit might be caused by the same fault is covered grossly by timing out an active history file if the elapsed time since the last interrupt exceeds some threshold, e.g., fifteen minutes. The premise is that most faults in the No. 4 ESS will stimulate an interrupt within that time period, until recovery is effected. Interrupts occurring at greater intervals have a high probability of independence. After repair has been made, and the unit appears fault free, the time-out interval should be sufficiently short to permit a subsequent fault in that unit to be treated with a new recovery sequence. When an active history file has been timed out, it no longer influences subsequent interrupts from a given unit. A record of the events up to that point are, however, kept in long-term storage for manual analysis, if needed.

The mobility provided by the escape mechanism, and to a lesser degree, by the time-out of active history files, permits the error analysis process to cope with possible inaccuracies in resolution, changing fault signatures, and the occurrences of apparently disassociated events.

VI. EXPERIENCE AND EVALUATION

To date, a total of 38 different recovery strategies have been developed for the No. 4 ESS peripheral maintenance system to handle a wide range of fault conditions and software problems involving more than a dozen types of units. Fifteen of these strategies serve the common needs of several units, and twenty three are specialized recovery sequences for particular units or for unique fault situations.

New recovery sequences are developed for each new unit or important new feature added to an existing unit. Recovery sequences might require changes as units are redesigned for cost reduction. System exposure and field support activities provide important feedback to the ongoing process of evaluating and improving the performance of the recovery system.

6.1 Adaptability

A very important attribute to build into the design of a fault-tolerant system is adaptability, i.e., the ability to change as experience and requirements dictate. The initial design of the recovery strategies and control programs in the No. 4 ESS did not adequately take into account the need to switch strategies in midstream of a recovery sequence. Alternate strategies frequently were not available or at least not specified by the isolation programs; and even when available, the escape control mechanism was distributed throughout the many recovery programs, and was difficult to change. Program changes were cumbersome to make, if not with high risk. Through experience and evolution, the No. 4 ESS peripheral fault recovery system has become highly structured and has matured into a system that has a high degree of adaptability. The escape data allows the error analysis process to either escalate a previous strategy, or to discontinue the previous and invoke a new strategy, as successive attempts are made at recovery. This feature has given the No. 4 ESS the flexibility to take full advantage of experience.

Strategies are developed based on a knowledge of system requirements and predicated on an expected system behavior under a given fault condition. Frequently, however, one learns that when the hardware and software are brought together for testing, the system behavior is not exactly as expected. Whether the cause is the accuracy of detection or accuracy of resolution, or a combination of both, adaptability becomes essential. This is particularly important when recovery problems are exposed in a software/hardware package deployed in switching systems carrying live traffic. It must be possible to develop and test program changes and deploy them in the field as quickly as possible, with minimum risk to the quality of service and to the integrity of the overall system. This has been a major goal in the development of the fault-tolerant No. 4 ESS switching system.

6.2 Experience

The performance of a fault tolerant system can be measured by its track record in encounters with hardware failures and other system troubles.

Software deficiencies and coding errors will always exist in a large system. It is not cost-effective and probably not possible to test a system to the point where all problems have been found and removed. What is important is that service-affecting incidents are kept to a minimum, and that any such occurrences are analyzed for cause. Program corrections or enhancements are developed thereupon, when appropriate.

The level of fault tolerance demanded for the No. 4 ESS is to detect a failure and isolate and recover from that failure with minimum disturbance to call processing. A well-behaved hard fault, i.e., one that reacts consistently under test, can generally be isolated on a single stimulus, and recovery can be effected immediately if redundancy is available. Transients, or other faults that cannot be uniquely resolved, can reasonably require two or more interrupts before they are successfully isolated. These are reasonable levels of performance.

If a fault occurs at a time when the redundant unit is not available, i.e., when the mate unit is out of service either for routine testing or an actual fault, the current fault represents a duplex failure situation. The resolution may be clear, but the recovery cannot be executed without disrupting service. Under these circumstances, an attempt is made to tolerate the problem as long as possible. That is the purpose of the leaky bucket strategy, which estimates how well the system can tolerate the fault by measuring the frequency of the interrupts. A low rate of interrupt can be tolerated longer than a high rate. The actual attempt at recovery will occur at the point that the leaky bucket threshold is exceeded. This action will generally have a disruptive effect on service. The number of interrupts required to reach that point is, of course, a function of the rate of the interrupts.

Since the introduction of the No. 4 ESS in January 1976, seven program generics or versions have been developed and put into general service in more than 75 offices. While the primary impetus for a generic development has been to add new features or major cost reductions, many improvements have been introduced in the recovery system in each generic, in the form of corrections or design enhancements. Most of these were stimulated through laboratory testing, and through feedback from the field. Threshold parameters have been fine tuned in response to field experiences. Several strategies were made more tolerant of faults, and others were rendered more sensitive.

The principal goal of the No. 4 ESS fault-tolerant system is to recover from any fault with minimum impact on service and with a minimum number of interrupts. When recovery is possible, no strategy should inadvertently cause service degradation, e.g., any unnecessary zero-start or duplex failure actions. All service-affecting incidents are analyzed for cause by a field support team. Of the total number of incidents causing service outages for the period January through June 1981, less than ten percent were judged to be caused by deficiencies in the peripheral maintenance software system being discussed herein, whereas hardware problems and procedural errors accounted for almost 50 percent of the incidents.

VII. SUMMARY

The fault-tolerant design of the No. 4 ESS has been evolving since its introduction. Sophisticated recovery strategies have been developed to give the system the necessary high level of fault tolerance. The incident and recovery data printouts have been designed to assist the support team in making analysis of a problem fast and accurate. Fault isolation can implicate multiple units in some situations where a unique resolution is not possible. Multiple recovery actions can be carried out on a single stimulus.

At the current state of the art, the No. 4 ESS is a high-performance switching system, with a successful record for reliability and availability. Its fault-tolerant design has contributed significantly to that success.

REFERENCES

1. J. J. Kulzer, "Systems Reliability—A Case Study for No. 4 ESS," INFOTECH State of the Art Conference on Computer System Reliability, London, England, June 1977.
2. M. N. Meyers, W. A. Routt, and K. W. Yoder, "Maintenance Software," B.S.T.J., *56,* No. 7 (September 1977), pp. 1139-67.
3. P. K. Giloth and H. E. Vaughan, "Early No. 4 ESS Field Experience," Int. Switching Symp., Kyoto, Japan, October 1976, pp. 241-4-1-7.
4. P. K. Giloth, "No. 4 ESS Reliability and Maintainability Experience," 1980 Proc. Ann. Reliability and Maintainability Symp., San Francisco, CA, January 22–24, 1980, pp. 388-92.

CONTRIBUTORS TO THIS ISSUE

Michael K. Brown, B.S.E.E., 1973, M.S., 1977, and Ph.D., 1981 (Electrical Engineering), all from the University of Michigan, Ann Arbor; Burrough's Corp., 1973-1980; Bell Laboratories, 1980—. From 1973 to 1976 Mr. Brown was employed with the Burrough's Corp. and was involved in the development of ink jet printer systems. From 1976 to 1980 he continued his work with Burrough's Corp. as a consultant while pursuing his Ph.D. degree at the University of Michigan. His thesis was in the area of image processing and pattern recognition. Since 1980, he has been with the Speech Processing Group at Bell Laboratories, Murray Hill, NJ, where he has been involved in research in speech recognition and synthesis techniques.

Paul J. Burke, B.S. (Mathematics), 1940, City College of New York; Ed.M. (Educational Measurement), 1950, Harvard University; Ph.D. (Mathematical Statistics), 1966, Columbia University; Bell Laboratories, 1953—. Mr. Burke's work is in the field of telephone traffic theory and its applications. Member, Phi Beta Kappa, Sigma Xi, Operations Research Society of America; Fellow, AAAS.

Kai Y. Eng, B.S.E.E. (summa cum laude), 1974, Newark College of Engineering; M.S. (Electrical Engineering), 1976, Dr. Engr. Sc. (Electrical Engineering), 1979, Columbia University; RCA Astro-Electronics, 1974–1979; Bell Laboratories, 1979—. Mr. Eng has worked on various areas of microwave transmission, spacecraft antenna analysis, and communications satellites. He is presently a member of the Radio Research Laboratory, studying TV transmission through satellites. Member, IEEE, Sigma Xi, Tau Beta Pi, Eta Kappa Nu, Phi Eta Sigma.

Stephen R. Forrest, B.A. (Physics), 1972, University of California, Berkeley; M.S., and Ph.D. (Physics), 1974 and 1979, respectively, University of Michigan, Ann Arbor. From 1972 to 1973 Mr. Forrest worked as a Semiconductor Process Engineer at Sierra Electronics, a company operated at that time by Philco-Ford in Menlo Park, CA. He was involved in the characterization and design of long-wavelength p-i-n and avalanche photodiodes for Bell Laboratories, Murray Hill, NJ. Currently, he is Supervisor of the Integrated Opto-Electronic Devices and Circuits Group at Murray Hill.

Barry G. Haskell, B.S. (Electrical Engineering), 1964, M.S., 1965, and Ph.D., 1968, University of California, Berkeley; University of California, 1965-1968; Bell Laboratories, 1968—; Rutgers University, 1977–1979. Mr. Haskell was a Research Assistant at the University of California Electronics Research Laboratory and a part-time faculty member of the Department of Electrical Engineering at Rutgers University. At Bell Laboratories, he is presently Head of the Radio Communications Research Department, where his research interests include television picture coding and transmission of digital and analog information via microwave radio. Member, IEEE, Phi Beta Kappa, Sigma Xi.

Wendy Keilin, Massachusetts Institute of Technology (Electrical Engineering), 1980—; Bell Laboratories, 1980—. Ms. Keilin has worked in the Acoustics Research Department under the Bell Laboratories Engineering Scholarship Program. She is currently working with the CMOS Integrated Circuit Department studying hot electron injection in thin gate oxides.

Ytzhak Levendel, B.S.E.E., 1971, Technion-Israel; M.S.C.S., 1974, The Weitzman Institute of Science; Ph.D., 1976, University of Southern California; Bell Laboratories, 1976—. Mr. Levendel has done research in fault diagnosis and until lately was involved in the development of a logic and test design aid system. He is now a Supervisor in No. 5 ESS. Member, IEEE, Eta Kappa Nu.

Premachandran R. Menon, B.S. (Electrical Engineering), 1954, Banaras Hindu University; Ph.D. (Electrical Engineering), 1962, University of Washington; Bell Laboratories, 1963—. Mr. Menon has done research in switching theory and fault diagnosis and is currently involved in the development of a logic simulation system. Recipient of the Distinguished Technical Staff Award; member, IEEE.

Debasis Mitra, B.Sc., 1965, and Ph.D., 1967 (Electrical Engineering), London University; United Kingdom Atomic Energy Authority Research Fellow, 1966–1967; Bell Laboratories, 1968—. Mr. Mitra has worked on the stability analysis of nonlinear systems, semiconductor networks, growth models for new communication systems, speech waveform coding, nonlinear phenomenon in digital signal processing, adaptive filtering, and network synchronization. Most recently, he has been involved in the analytic and computational aspects of stochastic networks and computer communications. He is a Supervisor in the Mathematics of Physics and Networks Department. Member, IEEE, SIAM.

Suresh H. Patel, B.E. (Electrical), 1975, University of Zambia; M.S.E.E., 1976, Illinois Institute of Technology; Association of American Railroads, 1977-1981; Bell Laboratories, 1981—. At the Association of American Railroads Mr. Patel was involved in hardware/software design, development and testing of a multi-microcomputer-based instrumentation system for freight trains. He was also involved in development of analytical and computer models on track circuit. At Bell Laboratories he is a member of the Processor Design Department. Mr. Patel was a part-time instructor at Illinois Institute of Technology during 1977-1978.

Lawrence R. Rabiner, S.B. and S.M., 1964, Ph.D. (Electrical Engineering), The Massachusetts Institute of Technology: Bell Laboratories, 1962—. From 1962 through 1964, Mr. Rabiner participated in the cooperative plan in electrical engineering at Bell Laboratories. He worked on digital circuitry, military communications problems, and problems in binaural hearing. Presently, he is engaged in research on speech communications and digital signal processing techniques. He is coauthor of *Theory and Application of Digital Signal Processing* (Prentice-Hall, 1975) and *Digital Processing of Speech Signals* (Prentice-Hall, 1978). Former President, IEEE, ASSP Society; former Associate Editor, ASSP Transactions; former member, Technical Committee on Speech Communication of the Acoustical Society, ASSP Technical Committee on Speech Communication; Member, IEEE Proceedings Editorial Board, Eta Kappa Nu, Sigma Xi, Tau Beta Pi. Fellow, Acoustical Society of America, IEEE.

K. G. Ramakrishnan, B.S., 1970, M.S., 1972 (Electrical Engineering), Indian Institute of Technology, Kanpur, India; M.S., 1976, Ph.D., 1978 (Computer Science), Washington State University; Bell Laboratories, 1978—. Mr. Ramakrishnan has worked on developing analytical and simulation tools for the Advanced Communication Service project. He is currently involved in research on distributed simulation, techniques for code improvement, and queueing network packages. Member ORSA, SIAM, ACM.

Aaron E. Rosenberg, S.B. (E.E.) and S.M. (E.E.) 1960, Massachusetts Institute of Technology, Ph.D. (E.E.), 1964, University of Pennsylvania; Bell Laboratories, 1964—. Mr. Rosenberg is presently engaged in studies of systems for man-machine communication-by-voice in the Acoustics Research Department at Bell Laboratories. Member, Eta Kappa Nu, Tau Beta Pi, Sigma Xi; Fellow, Acoustical Society of America; Member, IEEE and IEEE Acoustics, Speech and Signal Processing Group Technical Committee on Speech Processing; Associate editor for speech processing, IEEE Transactions on Acoustics, Speech and Signal Processing.

R. G. Smith, B.S., 1958, M.S., 1959, and Ph.D., 1963, Stanford University; Bell Laboratories, 1963—. Mr. Smith has worked on Nd:YAG lasers, optical second harmonic generation, and optical parametric oscillators. From 1968 to 1982 he supervised groups responsible for the development of nonlinear optical devices and, more recently, detectors and receivers for lightwave systems. He is currently Head of the Lightwave Transmitters Department. Past President, Quantum Electronics and Applications Society; Co-Chairman, CLEOS '80, and Chairman, CLEO '81 Steering Committee. Member, Optical Society of America, The American Physical Society, Phi Beta Kappa, Tau Beta Pi.

Raymond Steele, (SM '80), B.S. (Electrical Engineering) from Durham University, Durham, England, in 1959, and the Ph.D. degree in 1975. Prior to his enrollment at Durham University, he was an indentured apprenticed Radio Engineer. After research and development posts at E. K. Cole Ltd., Cossor Radar and Electronics, Ltd., and The Marconi Company, all in Essex, England, he joined the lecturing staff at the Royal Naval College, Greenwich, London, England. Here he lectured in telecommunications to NATO and the External London University degree courses. His next post was as Senior Lecturer in the Electronic and Electrical Engineering Department of Loughborough University, Loughborough, Leics., England, where he directed a research group in digital encoding of speech and television signals. In 1975 his book, *Delta Modulation Systems* (New York: Halsted), was published. He was a consultant to the Acoustics Research Department at Bell Laboratories in the summers of 1975, 1977, and 1978, and in 1979 he joined the company's Communications Methods Research Department, Crawford Hill Laboratory, Holmdel, NJ.

Uzi Timor, B.S. (Electrical Engineering), 1957, M.S. (Electrical Engineering), 1963, Technion, Israel Institute of Technology, Haifa, Israel; Ph.D., 1969, University of California, Berkeley; Armament Development Authority/Israel Ministry of Defence, 1959-1966, 1972—; Member of the Technical Staff, Jet Propulsion Laboratory, Pasadena, California, 1969-1972; Adjunct Faculty (Electrical Engineering), Technion, Israel Institute of Technology, Haifa, Israel, 1972-1980; Bell Laboratories, 1979-1980. At Jet Propulsion Laboratory, Mr. Timor was engaged in research in digital communications and coding. As a consultant at Bell Laboratories, he worked on digital communications and mobile radio. Member, IEEE, Eta Kappa Nu.

Robert J. Willett, B.S.E.E., 1954, University of Maine; M.S.E.E., 1960, New York University; Bell Laboratories, 1954—. Mr. Willett has worked on circuit designs for the Titan Missile Guidance System and a forward-looking missile project, memory design for the UNICOM project, and program development for the AUTOVON project, in the areas of call processing, terminal maintenance, and peripheral maintenance. He has also been involved in the development of the No. 4 ESS peripheral system, and the peripheral maintenance programs, particularly the craft interface control system, fault recovery, and error analysis. His current assignment is in the Software System Design and Planning Group of No. 5 ESS. He holds a patent (co-inventor) on a fault-detection arrangement for digital transmission system used in the No. 4 ESS. In July 1982 Mr. Willett received a Bell Laboratories Distinguished Technical Staff award. Member, Tau Beta Pi, Phi Kappa Phi.

Jay G. Wilpon, B.S. and A.B. (cum laude) in Mathematics and Economics, respectively, from Lafayette College, Easton, Pa., 1977; M.S. (Computer Science), 1982, Stevens Institute of Technology, Hoboken, N.J.; Bell Laboratories, 1977—. Since June 1977 Mr. Wilpon has been with the Acoustics Research Department at Bell Laboratories, Murray Hill, N.J., where he is a Member of the Technical Staff. He has been engaged in speech communications research and is presently concentrating on problems of speech recognition.

Wai Choong Wong was born on June 9, 1954, in Ipoh, Malaysia. He received the B.Sc. degree in Electronic and Electrical Engineering and the Ph.D. degree in Electronic Engineering both from Loughborough University of Technology, Loughborough, England, in 1976 and 1980, respectively. Since April 1980 he has been a Member of Technical Staff in the Communications Methods Research Department of Bell Laboratories at Crawford Hill, Holmdel, New Jersey. His current interest is in speech signal processing, particularly applied to mobile radio environment, digital modulation techniques, and simultaneous transmission of data and speech signals.

Costas S. Xydeas was born in Piraeus, Greece, in 1950. He received the first degree in Electronic Engineering from Vranas Higher School of Electronics, Athens, Greece, in 1972, and the M.Sc. and Ph.D. degrees in Electrical Engineering from Loughborough University of Technology, Loughborough, Leics., England, in 1974 and 1978, respectively. In 1977 he joined the Department of Electrical Engineering of Loughborough University of Technology as a Research Fellow and was engaged in research on low-bit-rate digitization of speech signals. He is presently a Lecturer at Loughborough University of Technology, where he is directing a research group in digital coding and processing of speech signals. His areas of interest include adaptive linear and nonlinear prediction, adaptive quantization, low-bit-rate coding of speech, suppression of acoustic noise in speech, enhancement of band-limited speech, and encryption of digitized speech signals.

PAPERS BY BELL LABORATORIES AUTHORS

COMPUTING/MATHEMATICS

Conway J. H., Sloane N. J. A., **On the Enumeration of Lattices of Determinant One.** J Number Th 15(1):83-94, 1982.

Fishburn P. C., **Relative Probabilities of Majority Winners Under Partial Information.** Math Soc Sc 3(1):73-78, 1982.

Mallows G. L., Vardi, Y., **Bounds on the Efficiency of Estimates Based on Overlapping Data.** Biometrika 69(2):287-296, 1982.

McIlroy, M. D., **The Number of States of a Dynamic Storage-Allocation System.** Computer J 25(3):388-392, 1982.

ELECTRICAL ENGINEERING

Brainard R. C., Netravali A. N., Pearson D. E., **Composite Television Coding—Subsampling and Interpolation.** Smpte J 91(8):717-724, 1982.

Chen C. Y., Cho A. Y., Alavi K., Garbinski P. A., **Short Channel $Ga_{0.47}In_{0.53}As$/ $Al_{0.48}In_{0.52}As$ Selectively Doped Field-Effect Transistors.** Elec Dev L 3(8):205-208, 1982.

Cordell R. R., **A New Family of Active Variable Equalizers.** IEEE Circ S 29(5):316-322, 1982.

Farrow R. C., Joy D. C., **A New Technique for Electron Channeling Pattern Measurements.** Scan Elec M1981(P1):397-402, 1981.

Feldman L. C., **Comparison of High-Energy Ion-Beam and Electron-Beam Surface Probes.** Appl Surf S 13(1-2):211-230, 1982.

Fischer J. H., **Noise Sources and Calculation Techniques for Switched Capacitor Filters.** IEEE J Soli 17(4):742-752, 1982.

Fleming R. M., **The Non-Linear Response of NbSe3 to Pulsed Electric-Fields.** Sol St Comm 43(3):167-170, 1982.

Joy D. C., Newbury D. E., Davidson D. L., **Electron Channeling Patterns in the Scanning Electron-Microscope.** (Review or Bibliog.) J Appl Phys 53(8):R81-R122, 1982.

Kurth C. F., Bures K. J., Gagnon P. R., Etzel M. H., **A Per-Channel, Memory-Oriented Transmultiplexer with Logarithmic Processing.** IEEE Commun 30(7):1520-1527, 1982.

Liebesma B. S., Tortorella M., **Reliability of a Class of Telephone Switching Systems.** P An Rel M 1982(NSYM):120-124, 1982.

Mammel W. L., Cohen L. G., Lumish S., **Improving Propagation Characteristics in Single-Mode Optical Fibers with Computer-Aided Analysis Using Wave-Equation Techniques.** P Soc Photo 294:26-33, 1981.

Marshall T. G., **A Multiple VLSI Signal Processor Realization of a Transmultiplexer.** IEEE Commun 30(7):1560-1568, 1982.

Meyers M. H., **A Reduced Complexity Moment Algorithm for Correlated Digital Signals (Letter).** IEEE Commun 30(7):1798-1799, 1982.

Mounce A., **Computationally Efficient Determination of D/A, and A/D Noise in Digital Transmultiplexers.** IEEE Commun 30(7):1477-1482, 1982.

Rummler W. D., **A Comparison of Calculated and Observed Performance of Digital Radio in the Presence of Interference.** IEEE Commun 30(7):1693-1700, 1982.

Swartz R. G., Wooley B. A., Voshchenkov A. M., Archer V. D., Chin G. M., **An Integrated-Circuit for Multiplexing and Driving Injection-Lasers.** IEEE J Soli 17(4):753-760, 1982.

Tamari N., **Growth and Characterization of Cd-Doped InGaAsP/InP Double Heterostructure Lasers.** J Elec Mat 11(4):611-629, 1982.

Wyatt J. L., Chua L. O., Gannett J. W., Goknar I. C., Green D. N., **Energy Concepts in the State-Space Theory of Non-Linear Normal-Ports 2 Losslessness.** IEEE Circ S 29(7):417–430, 1982.

MANAGEMENT/ECONOMICS

Raj B., Vinod H. D., **Bell System Scale Economies Estimated From a Random-Coefficients Model.** J Econ Bus 34(3):247-252, 1982.

PHYSICAL SCIENCES

Aspnes D. E., **Imaging Performance of Mirror Pairs for Grazing-Incidence Applications—A Comparison.** Appl Optics 21(14):2642-2646, 1982.

Aspnes D. E., Kelso S. M., **Properties and Performance of Grazing-Incidence Reflectors.** P Soc Photo 315:30-36, 1981.

Bishop D. J., Dynes R. C., Tsui D. C., **Magnetoresistance in Si Metal-Oxide-Semiconductor Field-Effect Transistors—Evidence of Weak Localization and Correlation.** Phys Rev B 26(2):773–779, 1982.

Bosch M. A., **Hydrogenated Semiconductors as Optical-Recording Media.** P Soc Photo 329:181–185, 1982.

Brinkman W. F., Fisher D. S., Moncton D. E. **Melting of 2-Dimensional Solids.** Science 217(4561):693–700, 1982.

Cheng J., **Homogeneous Electric-Field Threshold Switching Phenomena in a Multiplexible Bistable Liquid-Crystal Display.** J Appl Phys 53(8):5584–5595, 1982.

Craighead H. G., Howard R. E., Smith R. W., Snyder D. A., **Textured Optical Storage Media.** P Soc Photo 329:202–205, 1982.

Dahlberg S. C., Reinganum C. B., **Elovich Photo-Voltage Decay Kinetics Observed for the Organic Semiconductor 3,4,9,10-Perylenetetracarboxylic Dianhydride** (Letter). Surf Sci 119(1):L326–L330, 1982.

Dean B. A., Dixon M., **Functional Life Testing of Multi-Moded Lasers for Digital-Communications Applications.** P Soc Photo 328:35–41, 1982.

Dubois L. H., Schwartz G. P., **Surface Optical Phonons and Hydrogen Chemisorptions on Polar and Non-Polar Faces of GaAs, InP, and GaP.** Phys Rev B 26(2):794–802, 1982.

Feldman L. C., Poate J. M., **Rutherford Backscattering and Channeling Analysis of Interfaces and Epitaxial Structures (Review or Bibliog.).** Ann R Mater 12:149–176, 1982.

Feldman M., White A. D., White D. L., **Application of Zone Plates to Alignment in X-Ray-Lithography.** P Soc Photo 333:124–130, 1982.

Forrest S. R., Kim O. K., **Deep Levels in $In_{0.53}$ $Ga_{0.47}$ As/InP Heterostructures.** J Appl Phys 53(8):5738–5745, 1982.

Fuls E. N., **X-Ray-Lithography Applied to the Fabrication of Mu-M N-Channel Metal-Oxide Semiconductor (NMOS) Circuits.** P Soc Photo 333:113–117, 1982.

Golovchenko J. A., Patel J. R., Kaplan D. R., Cowan P. L., Bedzyk M. J., **Solution to the Surface Registration Problem Using X-Ray Standing Waves.** Phys Rev L 49(8):560-563, 1982.

Gottscho R. A., Smolinsky G., Burton R. H., **Carbon-Tetrachloride Plasma-Etching of GaAs and InP—A Kinetic Study Utilizing Nonperturbative Optical Techniques.** J Appl Phys 53(8):5908–5919, 1982.

Gregori G. P., Lanzerotti L. J., Meloni A., **Geomagnetic Depth Sounding by Induction Arrow Representation—Reply (Letter).** Rev Geophys 20(3):523–528, 1982.

Griffiths J. E., **Dye-Laser Raman-Scattering in Bulk Selenium Glass.** Sol St Comm 43(4):253–255, 1982.

Hackwood S., Dayem A. H., Beni G., **Amorphous-Nonmetal to Crystalline-Metal Transition in Electrochromic Iridium Oxide-Films.** Phys Rev B 26(2):471–478, 1982.

Kelley D. F., Rentzepis P. M., **Predissociation and Picosecond Spectroscopy of Transient Species.** P Soc Photo 322:206–214, 1982.

Lang D. V., **Recombination-Enhanced Reactions in Semiconductors (Review or Bibliog.).** Ann R Mater 12:377–400, 1982.

Levin R. M., **Water-Absorption and Densification of Phosphosilicate Glass-Films.** J Elchem So 129(8):1765–1770, 1982.

Levyclement C., Heller A., Bonner W. A., Parkinson B. A., **Spontaneous Photoelectrolysis of HBr and HI.** J Elchem So 129(8):1701–1705, 1982.

Macrander A., Barr D., Wagner A., **Resist Possibilities and Limitations in Ion-Beam Lithography.** P Soc Photo 333:142–151, 1982.

Madden H. H., et al., **Correlation of Stimulated H^+-Desorption Threshold With Localized State Observed in Auger Line Shape—Si(100):H.** Phys Rev B 26(2):896–902, 1982.

Malm D. L., Riley J. E., **Characterization of Near-Surface Composition of Borosilicate Glasses (BSG) by Secondary Ion Mass-Spectrometry (SIMS).** J Elchem So 129(8):1819–1821, 1982.

Moerner W. E., Sievers A. J., Silsbee R. H., Chraplyvy A. R., Lambert D. K., **Persistent Holes in the Spectra of Localized Vibrational-Modes in Crystalline Solids.** Phys Rev L 49(6):398–401, 1982.

Nopper R. W., Hughes W. J., Maclennan C. G., Mcpherro R. L., **Impulse-Excited Pulsations During the July 29, 1977, Event.** J Geo R-S P 87(NA8):5911–5916, 1982.

Ondrias M. R., Rousseau D. L., Kitagawa T., Ikedasai M., Inubushi T., Yonetani T., **Quaternary Structure Changes in Iron-Cobalt Hybrid Hemoglobins Detected by Resonance Raman-Scattering.** J Biol Chem 257(15):8766–8770, 1982.

Serri J. A., Cardillo M. J., Becker G. E., **A Molecular-Beam Study of the NO Interaction with Pt(111).** J Chem Phys 77(4):2175–2189, 1982.

Smolinsky G., Mayer T. M., Truesdale E. A., **Plasma-Etching of Silicon and Silicon Dioxide with Hydrogen-Fluoride Mixtures.** J Elchem So 129(8):1770–1772, 1982.

Tai K. L., Vadimsky R. G., Ong E., **Multilevel Ge-Se Film Based Resist Systems.** P Soc Photo 333:32–39, 1982.

Tu C. W., Schlier A. R., **Surface Studies of InP by Electron-Energy Loss Spectroscopy and Auger-Electron Spectroscopy.** Appl Surf S11-2(JUL):355–361, 1982.

Venkatesan T., Wilkens B., **On the Mechanism of the Lithographic Sensitivity Enhancement of Obliquely Deposited Germanium Selenide Films.** P Soc Photo 333:163–167, 1982.

Voshchenkov A. M., Hanson R. C., **Submicron Resolution Photolithography by Spectral Shaping.** Elec Dev L 3(7):208–210, 1982.

Whangbo M. H., Cava R. J., Disalvo F. J., Fleming R. M. **The Electronic-Structure of $Fe_{1+x}Nb_{3-x}Se_{10}$.** Sol St Comm 43(4):277–282, 1982.

Whangbo M. H., Disalvo F. J., Fleming R. M., **Electronic-Structure of $ZrTe_5$.** Phys Rev B 26(2):687–689, 1982.

Yousuf M., Kuliyev B., Lalevic B., Poteat T. L., **Po-InP Schottky Diode Hydrogen Sensors.** Sol St Elec 25(8):753–758, 1982.

SOCIAL and LIFE SCIENCES

Egan D. E., **A Heuristic for Componential Analysis—Try Old Goals.** Behav Brain 5(2):348–350, 1982.

SPEECH/ACOUSTICS

Becker C. A., **The Development of Semantic Context Effects—2 Processes or 2 Strategies.** Read Res Q 17(4):482–502, 1982.

Umeda N., **F_0 Declination is Situation Dependent.** J Phonetics 10(3):279–290, 1982.

Wilpon J. G., Rabiner L. R., Bergh A., **Speaker-Independent Isolated Word Recognition Using a 129-Word Airline Vocabulary.** J Acoust So 72(2):390–396, 1982.

CONTENTS, JANUARY 1983

Bell System