

Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels

Javan Erfanian, *Member, IEEE*, Subbarayan Pasupathy, *Fellow, IEEE*,
and Glenn Gulak, *Member, IEEE*

Abstract — The problem of practical realization of the optimal fixed-delay symbol-by-symbol detection algorithm, which is optimum in the sense of minimizing the symbol error probability, given a delay constraint D , is investigated. A fully-parallel structure is developed, and through systematic reformulations of the algorithm, the computational requirements are reduced considerably. In addition, the problems associated with a large dynamic range such as overflow (or underflow) are (practically) removed. A number of approximations are applied to this simplified parallel symbol (SPS) detector that lead to the derivation of suboptimal detectors. One such suboptimal detector is shown to be the same as the minimum-metric Viterbi detector. A brief comparison of the SPS detector and the Viterbi detector shows that the former has a slightly better performance at low values of signal-to-noise ratio (SNR) and the latter performs a smaller number of computations (particularly) at higher values of SNR; otherwise, the two detectors are comparable in performance and complexity.

I. INTRODUCTION

An increasing demand for high data rate transmission over bandlimited channels with severe intersymbol interference (ISI) has resulted in a flurry of activity over the last two decades to develop improved methods of equalization [1]. Since error probability is the most important performance measure in such applications, it is of interest to develop practical VLSI implementable receivers which optimize some measures directly related to probability of error. Two important criteria in this regard are: maximum likelihood *sequence* estimation (MLSE) and minimum probability of *symbol* error.

An important breakthrough came with the derivation of maximum-likelihood receivers for channels with *finite* memory. Taking advantage of the fact that the effective range of ISI is actually finite, Chang and Hancock derived a sequential procedure [2], in which the number of computations increased only linearly (rather than exponentially) with the message length. Following [2], Abend and Fritchman [3] derived the optimum fixed-delay 'symbol-by-symbol' detector, which is optimum in the sense of minimizing the symbol error probability given a fixed delay constraint, D (i.e. given all the previous as well as D succeeding received samples). This is

Paper approved by C.P. Jeremy Tzeng, the Editor for VLSI in Communications of the IEEE Communications Society. Manuscript received October 1, 1990 and revised November 15, 1991. The research was supported by grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada. Some of the results have been reported in IEEE Pacific Rim Conf. on Commun., Computers, & Signal Processing, Victoria, B.C., Canada, June 89, Canadian Conf. on Elec. & Computer Engg., Montreal, Canada, Sept. 89, and GLOBECOM'90, San Diego, Dec. 90.

Authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, M5S 1A4, Canada.

IEEE Log Number 9401568.

simpler than the optimum compound detector [4,5], in that the symbol estimation can be carried out before the entire data sequence is received. Nevertheless, neither this nor related simplifications, as in [6], were thought of as practical procedures and researchers have pointed out the reasons [7-9]. Unlike optimal symbol detection, the procedure that has received tremendous attention in the last two decades is the Viterbi algorithm (VA), which was originally introduced in 1967 for maximum-likelihood decoding of convolutional codes [10], and, after its optimality in the sense of MLSE was realized, it was applied to signaling on ISI channels [8,11,12].

While the structural aspects of MLSE (or the related VA) have received a great deal of attention [13-18], the criterion and/or algorithms based on the minimum probability of symbol error have not received the same scrutiny. The main purpose of the present paper is to examine an algorithm which minimizes symbol error probability with the objective of developing parallel-structure implementations with low complexity. We focus on the practical realization of the optimal fixed-delay symbol-by-symbol detection algorithm derived in [3]. First, the algorithm is mapped onto a fully-parallel structure suitable for VLSI implementation. Then, a number of simplifications are introduced, through systematic reformulations of the algorithm, that avoid the computation of exponentials and reduce (or possibly eliminate) the number of multiplications to be performed. All this is achieved at a price which seems acceptable in practice. In Section IV, a number of suboptimal design considerations are discussed and a few suboptimal symbol-by-symbol detectors are introduced. In Section V, the simplified parallel symbol (SPS) detector is compared to the Viterbi detector, where it is shown that a suboptimal SPS detector is identical to the minimum-metric Viterbi detector.

II. OPTIMAL SYMBOL-BY-SYMBOL DETECTION

A. The System Model

Figure 1 shows the block diagram of a digital transmission system with pulse amplitude modulation (PAM). The cascade of the transmitting filter, the channel filter, the receiver's whitening matched filter, and the sampler (Figure 1) can be represented by an equivalent discrete-time transversal filter with tap coefficients $\{f_k\}$ [7, p.355]. Then, the output sequence $\{v_k\}$ is given by

$$v_k = \sum_{n=0}^L f_n I_{k-n} + \eta_k \quad (1)$$

where $\{I_k\}$ is the sequence of information symbols which can

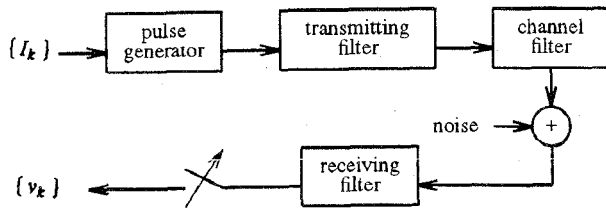


Fig. 1. Baseband digital transmission system.

take on any value \hat{I}_k drawn from an M -symbol alphabet ($\hat{I}_k = \pm 1$ in the binary case), L is the effective length of the channel dispersion, and $\{\eta_k\}$ are independent, identically distributed Gaussian random variables with zero mean and variance $N_0/2$.

The specific problem is to provide a delayed estimate of the transmitted symbol I_{k-D} , given the observed received sequence v_1, v_2, \dots, v_k , where D is the chosen delay constraint, according to some optimality criterion.

B. The Algorithm

We consider the optimal symbol-by-symbol detection algorithm under a fixed delay constraint, developed by Abend and Fritchman [3] and as presented in [7]. The algorithm is optimum in the sense of minimizing the probability of a symbol error, in comparison with all detectors which depend on the same number of received samples.

Let v_1, v_2, \dots, v_k be the observed received sequence, where $k > D \geq L$.¹ As is well known, minimization of symbol error probability is equivalent to MAP estimation. Hence, the algorithm computes the *a posteriori* probabilities $P(I_{k-D} = \hat{I}_{k-D} | v_k, \dots, v_1)$ for the M possible symbol values and chooses the one with the largest probability [7, p.387].

The MAP estimate of the information symbol at time $k-D$, denoted \tilde{I}_{k-D} , is given by

$$\tilde{I}_{k-D} = \arg \left\{ \max_{\hat{I}_{k-D}} \sum_{\hat{I}_{k-1}} \dots \sum_{\hat{I}_{k-D-1}} U_k(\hat{I}_k, \dots, \hat{I}_{k-D}) \right\} \quad (2)$$

In (2), the expression $\arg \{ G(\cdot) \}$ is equal to the value of \hat{I}_{k-D} that satisfies the function $G(\cdot)$, the term $\sum_{\hat{I}_i}(\cdot)$ is the summation over all possible values of \hat{I}_i , and the function $U_k(\cdot)$ is obtained recursively from

$$U_k(\hat{I}_k, \dots, \hat{I}_{k-D}) = p(v_k | \hat{I}_k, \dots, \hat{I}_{k-L}) P(I_k) \times \sum_{\hat{I}_{k-D-1}} U_{k-1}(\hat{I}_{k-1}, \dots, \hat{I}_{k-D-1}) \quad (3)$$

where $P(I_k)$ is the *a priori* probability of the information symbol I_k and $p(\cdot)$ denotes the probability density function of the received sample conditioned on the possible values of the information symbols and is given by

$$p(v_k | \hat{I}_k, \dots, \hat{I}_{k-L}) = \left\{ \exp \left[-\left(v_k - \sum_{j=0}^L f_j \hat{I}_{k-j} \right)^2 / N_0 \right] \right\} / (\pi N_0)^{1/2} \quad (4)$$

The initial condition for the recursion (3) is

$$U_{D+1}(\hat{I}_{D+1}, \dots, \hat{I}_1) \equiv p(v_{D+1} | \hat{I}_{D+1}, \dots, \hat{I}_{D+1-L}) \dots \dots p(v_2 | \hat{I}_2, \hat{I}_1) p(v_1 | \hat{I}_1) \times P(I_{D+1}, \dots, I_1) \quad (5)$$

C. The Computational Complexity

Using (4), the recursion in (3) can be replaced, for statistically independent, equi-probable (i.e., $P(I_i) = P_I$) input data, by

$$U'_k(\hat{I}_k, \dots, \hat{I}_{k-D}) = U_k(\hat{I}_k, \dots, \hat{I}_{k-D}) \times (\sqrt{\pi N_0} / P_I)^k \\ = \exp \left[-\left(v_k - \sum_{j=0}^L f_j \hat{I}_{k-j} \right)^2 / N_0 \right] \\ \times \sum_{\hat{I}_{k-D-1}} U'_{k-1}(\hat{I}_{k-1}, \dots, \hat{I}_{k-D-1}) \quad (6)$$

Using (5), the initial value for the recursion in (6), is obtained from

$$U'_{D+1}(\hat{I}_{D+1}, \dots, \hat{I}_1) = \exp \left\{ -\sum_{k=1}^{D+1} \left(v_k - \sum_{j=0}^L f_j \hat{I}_{k-j} \right)^2 / N_0 \right\} \quad (7)$$

The MAP estimate of the information symbol, at time $k-D$, is then given by

$$\tilde{I}_{k-D} = \arg \left\{ \max_{\hat{I}_{k-D}} \sum_{\hat{I}_{k-1}} \dots \sum_{\hat{I}_{k-D-1}} U'_k(\hat{I}_k, \dots, \hat{I}_{k-D}) \right\} \quad (8)$$

For simplicity, we only consider $M = 2$ (binary signaling) in this paper. The results can be simply generalized to any alphabet size. In (6), for each received sample v_k (or, alternatively, for each symbol estimate), 2^{L+1} exponentials need to be computed. The exponentials cannot be replaced by their exponents, because they are added rather than multiplied. In addition, there is a total of $2^{D+1} + 2^{L+1}$ multiplications required for the estimation of each symbol, as dictated by (6). We have ignored the operations involved in obtaining $(v_k - \sum_{j=0}^L f_j \hat{I}_{k-j})^2$

for which there may be different ways of implementation including using a table look-up. There are also 2^D additions, as suggested by (6), and $2(2^D - 1)$ additions, as required by (8). Furthermore, there is one binary comparison needed in (8). The operations required for the estimation of each symbol are summarized in the first column of Table 1.

The intimidating computational burden of the above algorithm is apparent from Table 1. The procedure involved is particularly complicated due to the large number of the slow and composite operations of exponentiation and multiplication required for the estimation of each symbol. In addition, there is a large dynamic range associated with the computation of exponentials. In a typical floating-point representation, this

¹ The algorithm for the case $D < L$ (also given in [3]) is similar to the case $D \geq L$ presented in this paper, except that it considers L (rather than D) most recent symbols for the recursion.

leads to overflow (or underflow) problems. A practical realization of this algorithm, which is the focus of this paper, is not possible without challenging its computational burden.

TABLE 1
COMPLEXITY OF SYMBOL DETECTORS

Operation	Number per symbol estimation			
	A. & F.*	SPS	Suboptimal Detector	
			'A2'	'B'
Exponent'n	2Y	---	---	---
Multipl'n	2N + 2Y	---	---	---
Addit'n	3N - 2	2.5N + 2	3Y	2Y
Comparison	1	2N - 1	2Y - 1	2Y - 1
Look-up	---	N/2 + 2	Y	---
Surviv'g seq.	---	---	(D-L) × Y	(D-L) × Y

* The original symbol detector derived by Abend & Fritchman [3].

Note: $Y = 2^L$ and $N = 2^D$.

III. SIMPLIFIED PARALLEL SYMBOL DETECTOR

A. Parallel Structure

To exploit the structure inherent in the algorithm and to take full advantage of its parallelism and regularity, we follow a procedure (as in [19]) towards a design that is suitable for VLSI implementation [20-23]. First, the recursive algorithm is mapped onto a parallel array structure.

To simplify the notation in (6), let

$$X_j^k = U'_k(\hat{I}_k, \hat{I}_{k-1}, \dots, \hat{I}_{k-D}) \quad (9)$$

$$\mu_j^k = (v_k - \sum_{i=0}^L f_i \hat{I}_{k-i})^2 \quad (10)$$

In (9) and (10), the time index k is chosen to match the time index of the received data v_k and the index j is the decimal value, in the range $j = 0, 1, \dots, 2^{D+1} - 1$, corresponding to the binary representation of a specific path (or sequence) among the paths given by all possible values of the $D + 1$ most recent information symbols. At time k , the index j can be obtained from

$$j = \sum_{i=0}^D 2^i \langle \hat{I}_{k-i}^{(j)} \rangle$$

where the function $\langle x \rangle$ is equal to x for positive x and zero otherwise. The 2^{D+1} paths, at any given time k , correspond to all possible values of the set $\{\hat{I}_{k-D}, \hat{I}_{k-D+1}, \dots, \hat{I}_k\}$ (e.g. if $D = 1$, then $\{\hat{I}_k, \hat{I}_{k-1}\} = \{\pm 1, \pm 1\}$, and $j = 0, 1, 2, 3$).

Then, with $N \equiv 2^D$, the algorithm defined by (6)-(8) can be rewritten as follows.

$$X_j^k = (X_i^{k-1} + X_{i+N}^{k-1}) \times e^{-\mu_j^k/N_0} \quad (11a)$$

$$i = 0, 1, \dots, N-1, \quad j = 2i, 2i+1, \quad k > D+1$$

$$X_j^{D+1} = \exp \left\{ - \sum_{k=1}^{D+1} \mu_j^k / N_0 \right\} \quad (11b)$$

$$\tilde{I}_{k-D} = \arg \left\{ \max_{l=0} \left[\sum_{j=0}^{N-1} X_j^k, \sum_{j=N}^{2N-1} X_j^k \right] \right\} \quad (11c)$$

Note that the expression in brackets in (11a) is the same in the computation of both X_{2i}^k and X_{2i+1}^k . Let

$$Y_i^k = X_i^k + X_{i+N}^k, \quad i = 0, 1, \dots, N-1 \quad (12a)$$

Then, (11a) can be rewritten as

$$X_j^k = Y_{\lfloor j/2 \rfloor}^{k-1} \times e^{-\mu_j^k/N_0}, \quad j = 0, 1, \dots, 2N-1 \quad (12b)$$

where the floor function $\lfloor \alpha \rfloor$ denotes the greatest integer not greater than α .

To exploit the parallelism in the recursive algorithm expressions given by (12), we first consider the index space (i, k) of (12a), as in [24], shown in Figure 2(a). If one node in this index space is allocated to the computation of each Y_i^k and the dependencies between nodes in each discrete time step are represented by directed arcs, a dependence graph [19] (DG, from which a parallel structure can be obtained) can be derived for the symbol-by-symbol detection algorithm. It must be noted that, according to (12), Y_i^k (the output at each node) is only a function of two outputs ($Y_{\lfloor i/2 \rfloor}^{k-1}$ and $Y_{\lfloor (i+N)/2 \rfloor}^{k-1}$) at time $k-1$. Hence, the DG illustrated for the case $D = 2$ in Figure 2(b) represents the recursion defined by (12).

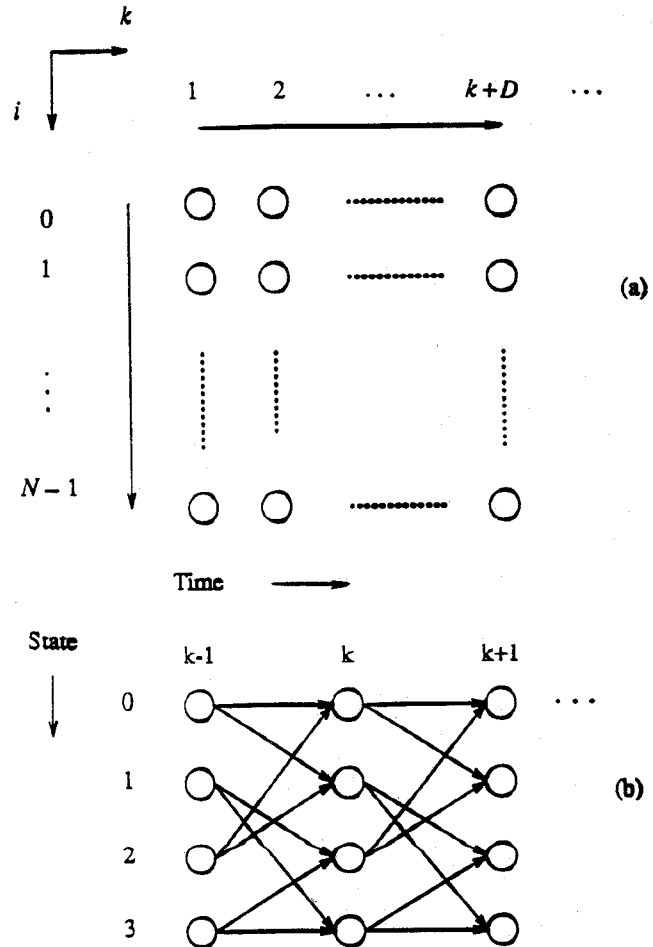


Fig. 2. Derivation of the dependence graph: (a) index space of (12) and (11c), (b) DG for the case $D = 2, N = 4$.

Let us define a *state* vector S_k , given by one set amongst all the possible values of the D most recent information symbols, as

$$S_k = (\hat{I}_{k-D+1}, \hat{I}_{k-D+2}, \dots, \hat{I}_k)$$

with $\hat{I}_k = 0$ for $k \leq 0$. Then, each node of the DG (at a particular time) describes a specific state, and the *breadth* of the DG consists of N states. Furthermore, since *all* the computations in each stage are based on *one* received sample v_k only, the number of stages in the *depth* of the DG is equal to the length of the data sequence.

In the index space (j, k) of the derived DG, (11c), which gives the optimal estimate of the k th information symbol, uses the intermediate node outputs $\{X_j^k\}$ given by (12b). According to (11c), detection of I_k depends on the comparison of the two sums (each of which could be interpreted as an average) $\sum_{j=0}^{N-1} X_j^{k+D}$ and $\sum_{j=N}^{2N-1} X_j^{k+D}$. Since $\{X_j^{k+D}\}$, for all j , are computed in parallel, no storage of path history is required.

It must be noted that, for equi-probable statistically independent binary information symbols, the algorithm given in (11c) and (12) is the same as the optimal fixed-delay symbol-by-symbol detection algorithm derived by Abend and Fritchman [3] and presented in (6)-(8). The latter is only regrouped to derive the fully-parallel array structure represented by the DG of Figure 2(b). An N -processor realization, one for each state, increases the throughput of the system by a factor of N compared to a serial implementation. Equation (12), processed at each node, is represented in Figure 3.

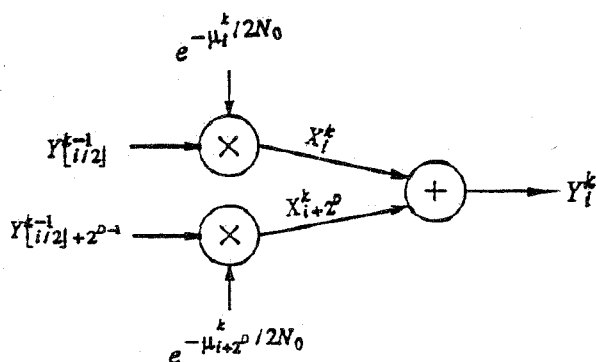


Fig. 3. Processing required at each node before simplification, (12).

B. A Low-Complexity Parallel Structure

The next step is to reduce the large computational burden of the algorithm. Parallel processing does increase the throughput significantly, but still, the computations required at each node are complex and slow, and are accompanied by other problems such as that of a large dynamic range. The exponential factors cannot be replaced by their exponents. Furthermore, the presence of ISI terms (in (7)) makes the exponents' dynamic range too large for the exponentials to be approximated by linear or quadratic expressions, even at low values of signal-to-noise ratio (SNR), where SNR is defined as the ratio of the average power of the signal to the average power of the noise in the signal bandwidth. For the binary

case considered here, $I_k = \pm 1$, the signal power is equal to 1, and therefore,

$$\text{SNR (dB)} = -10 \log_{10} N_0 / 2$$

Noting that X_i^k (11) and Y_i^k (12a) have positive values, let

$$x_i^k = -N_0 \ln X_i^k \quad \text{and} \quad y_i^k = -N_0 \ln Y_i^k \quad (13)$$

Without loss of generality, consider the case of $D=2$ (Figure 2(b)). To compute Y_0^k , the output at node $(0, k)$, state 0 at time k , (13) may be used in (12a) as follows.

$$e^{-y_0^k / N_0} = e^{-x_0^k / N_0} + e^{-x_4^k / N_0} \quad (14)$$

We now use the concept of *Jacobi's logarithm* to rewrite (14). The function Ω , called Jacobi's logarithm [25], is defined by the property

$$1 + a^\beta = a^{\Omega(\beta)}$$

where a is a primitive element in a finite field. By analogy,

$$e^{\beta_1} + e^{\beta_2} = e^{\beta_1} (1 + e^{\beta_2 - \beta_1}) = e^{\beta_1 + \ln(1 + e^{\beta_2 - \beta_1})}$$

Similarly,

$$e^{\beta_1} + e^{\beta_2} = \exp\{\max(\beta_1, \beta_2) + \ln(1 + e^{-|\beta_1 - \beta_2|})\} \quad (15)$$

Eqn.(14) can be rewritten as

$$\begin{aligned} e^{-y_0^k / N_0} &= \exp\{\max(-x_0^k / N_0, -x_4^k / N_0) + \ln(1 + e^{-|\delta_{0,4}^k| / N_0})\} \\ &= \exp\left\{\left[\min(x_0^k, x_4^k) - N_0 \ln(1 + e^{-|\delta_{0,4}^k| / N_0})\right] / (-N_0)\right\} \end{aligned} \quad (16)$$

where

$$\delta_{i,j}^k = x_i^k - x_j^k. \quad (17)$$

Let us use the following shorthand notation

$$\phi_{i,j}^k = -N_0 \ln(1 + e^{-|\delta_{i,j}^k| / N_0}) \quad (18)$$

Then, using (18), (16) is equivalent to

$$y_0^k = \min(x_0^k, x_4^k) + \phi_{0,4}^k$$

The recursive equations, for any given value of the delay constraint D , can now be obtained by generalizing the above simplification:

$$y_i^k = \min^*(x_i^k, x_{i+N}^k), \quad i=0, 1, \dots, N-1; k > D+1 \quad (19a)$$

where

$$\min^*(x_i^k, x_j^k) = \min(x_i^k, x_j^k) + \phi_{i,j}^k \quad (19b)$$

and $\phi_{i,j}^k$ is given by (18). Using (13) in (11b) and (12b) gives

$$x_i^{D+1} = \sum_{k=1}^{D+1} \mu_i^k, \quad i=0, 1, \dots, 2N-1 \quad (19c)$$

$$x_i^k = y_{[i/2]}^{k-1} + \mu_i^k, \quad i=0, 1, \dots, 2N-1; k > D+1 \quad (19d)$$

Note, from (18) and (19), that the knowledge of the noise variance, $N_0 / 2$, is now used in the computation of $\phi_{i,j}^k$ only. Also, from (18), $-N_0 \ln 2 \leq \phi_{i,j}^k < 0$, and is significant only for small values of $|\delta_{i,j}^k|$. This is further discussed in Section

IV. A significant simplification of the algorithm (19) occurs when only y_j^k is computed at every node (j, k) and the factor $\phi_{i,j}^k$ is tabulated explicitly as a function of $|\delta_{i+2^p}^k|$. Figure 4 shows the resulting simplification in the processing required at each node.

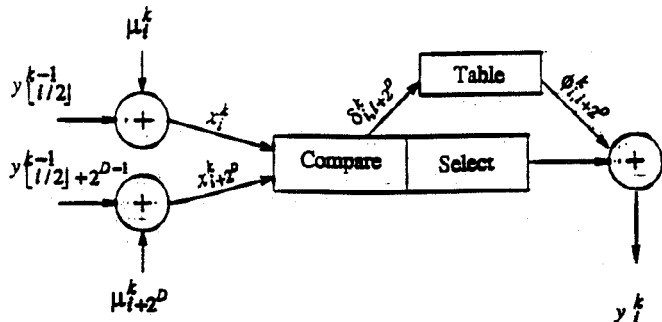


Fig. 4. Processing required at each node after simplification, (19).

Similarly, using (13a) in (11c) and applying the concept of Jacobi's logarithm, the MAP estimate of I_{k-D} , in the new structure, is obtained from

$$\tilde{I}_{k-D} = \arg \left\{ \min_{I_{k-D}} \left[\gamma_k(0), \gamma_k(1) \right] \right\} \quad (20)$$

where

$$\gamma_k(0) = \min^* (x_0^k, x_1^k, \dots, x_{N-1}^k) \quad (21a)$$

$$\gamma_k(1) = \min^* (x_N^k, x_{N+1}^k, \dots, x_{2N-1}^k) \quad (21b)$$

In Eqn.(21),

$$\min^* (a_0, a_1, \dots, a_{N-1}) \equiv \min (a_0, a_1, \dots, a_{N-1}) - N_0 \ln (e^{-\Delta_0/N_0} + e^{-\Delta_1/N_0} + \dots + e^{-\Delta_{N-1}/N_0}) \quad (22)$$

where

$$\Delta_i = a_i - \min (a_0, a_1, \dots, a_{N-1}), \quad i = 0, 1, \dots, N-1$$

Note that one of the $\{\Delta_i\}$ in (22) is zero and the remaining are positive, making one of the exponential terms in (22) equal to '1'. As Δ_i increases, $e^{-\Delta_i/N_0}$ in (22) decreases and becomes negligible compared to '1'. Consequently, a near-optimal approach could be to keep the two smallest Δ_i only, or, instead, if a 'tree' search is used in finding $\min(a_0, a_1, \dots, a_{N-1})$, to apply the table look-up only to the last binary comparison, as shown in (23). This way, the same look-up table used for the recursion may be used for output generation (symbol estimation). In this case, (21) may be approximated by

$$\gamma_k(0) \approx \min^* \left[\min(x_0^k, x_1^k, \dots, x_{N/2-1}^k), \min(x_{N/2}^k, x_{N/2+1}^k, \dots, x_{N-1}^k) \right] \quad (23a)$$

$$\gamma_k(1) \approx \min^* \left[\min(x_N^k, x_{N+1}^k, \dots, x_{3N/2-1}^k), \min(x_{3N/2}^k, x_{3N/2+1}^k, \dots, x_{2N-1}^k) \right] \quad (23b)$$

Equations (20) and (23), used for symbol estimation, involve $2N - 1$ binary comparisons, 2 look-up operations, and 2

additions. Note that if the optimal expressions (21) are used in the computation of $\gamma_k(0)$ and $\gamma_k(1)$, each binary comparison will be followed by one lookup-add operation. However, the approximation given by (23) is justified by the low sensitivity of the algorithm to the additive look-up term given by (18). This is further discussed in Section IV below.

Note (by comparing Figures 3 and 4) that *all* multiplications by $(-1/N_0)$ in the computation of $e^{-\Delta_i/N_0}$, and *all* exponentials are removed, and the multiplication inside every node is replaced by one addition, all at the expense of simple operations of compare-select and look-up (one per node) and an (affordable) extra complexity in symbol estimation ((20) replaces (11c)). The result is a simplified parallel symbol (SPS) detection algorithm (19)-(23) with a fully-parallel structure containing N nodes where the computations required at each node are simplified to add-compare-select followed by lookup-add (Figure 4), and where the storage of surviving sequences is not required. This algorithm is optimal if the symbol estimation given by (20) is performed using (21) and is near-optimal if (23) is used.

C. Further Simplification for $D > L$

By taking advantage of the fact that each state in the DG, derived in Section III.A, spans D information symbols whereas the terms $\{\mu_i^k\}$, given in (10), span only L information symbols, a further simplification is possible for the case $D > L$. Using (19d) in (19a), we get

$$y_i^k = \min^* \left[(y_{[i/2]}^{k-1} + \mu_i^k), (y_{[(i+N)/2]}^{k-1} + \mu_{i+N}^k) \right], \quad (24)$$

$$i = 0, 1, \dots, N-1; \quad k > D+1$$

But

$$\mu_i^k = \mu_{i+n \times 2^{L+1}}^k, \quad (25)$$

$$n = 0, 1, \dots, 2^{D-L}; \quad i = 0, 1, \dots, 2^{L+1} - 1$$

Therefore, using (25) and the identity

$$\min^* (A + C, B + C) = \min^* (A, B) + C \quad (26)$$

proved in Appendix A, (24) can be rewritten as

$$y_i^k = \min^* (y_{[i/2]}^{k-1}, y_{[(i+N)/2]}^{k-1}) + \mu_i^k, \quad (27)$$

$$i = 0, 1, \dots, N-1; \quad k > D+1$$

Let

$$z_i^k = \min^* (y_i^k, y_{i+N/2}^k), \quad i = 0, 1, \dots, N/2-1; \quad k > D \quad (28a)$$

Then, using (28a) in (27) gives

$$y_i^k = z_{[i/2]}^k + \mu_i^k, \quad i = 0, 1, \dots, N-1; \quad k > D \quad (28b)$$

Using (19) and (25), the initial conditions can now be obtained from

$$y_i^D = \sum_{k=1}^D \mu_i^k, \quad i = 0, 1, \dots, N-1 \quad (28c)$$

The MAP estimate of I_{k-D} is obtained from (20), as discussed in Section B above. Using (19d) in (21), we obtain

$$\gamma_k(0) = \min^* \left[(y_0^{k-1} + \mu_0^k, y_0^{k-1} + \mu_1^k), (y_1^{k-1} + \mu_2^k, y_1^{k-1} + \mu_3^k), \dots, (y_{N/2-1}^{k-1} + \mu_{N-2}^k, y_{N/2-1}^{k-1} + \mu_{N-1}^k) \right] \quad (29a)$$

$$\gamma_k(1) = \min^* \left[(y_{N/2}^{k-1} + \mu_N^k, y_{N/2}^{k-1} + \mu_{N+1}^k), (y_{N/2+1}^{k-1} + \mu_{N+2}^k, y_{N/2+1}^{k-1} + \mu_{N+3}^k), \dots, (y_{N-1}^{k-1} + \mu_{2N-2}^k, y_{N-1}^{k-1} + \mu_{2N-1}^k) \right] \quad (29b)$$

Using (25), (26), and the identity

$$\min^*(A, B, C, D) = \min^* \left[\min^*(A, B), \min^*(C, D) \right] \quad (30)$$

also proved in Appendix A, (29) can be rewritten as

$$\gamma_k(0) = \min^* \left[y_0^{k-1} + \min^*(\mu_0^k, \mu_1^k), y_1^{k-1} + \min^*(\mu_2^k, \mu_3^k), \dots, y_{N/2-1}^{k-1} + \min^*(\mu_{N-2}^k, \mu_{N-1}^k) \right] \quad (31a)$$

$$\gamma_k(1) = \min^* \left[y_{N/2}^{k-1} + \min^*(\mu_0^k, \mu_1^k), y_{N/2+1}^{k-1} + \min^*(\mu_2^k, \mu_3^k), \dots, y_{N-1}^{k-1} + \min^*(\mu_{N-2}^k, \mu_{N-1}^k) \right] \quad (31b)$$

The optimal SPS detection algorithm, given in (19)-(22) can be replaced by (20), (28), and (31) for the case $D > L$. The recursive equations (19) are replaced by (28), in which the number of computations are reduced by a factor of 2, and (21), used in (20) for MAP symbol estimation, are replaced by the simpler equations (31). The simplified algorithm can be represented by a DG, in which the nodes output $\{z_i^k\}$, given in (28), with the same topology as that derived earlier in this paper but with 2^{D-1} nodes (instead of 2^D nodes). It must be noted that insofar as the recursion in (28) is concerned, the DG could be further collapsed to one with 2^L nodes. However, this is restricted by (20) and (31), in which the $\{y_i^k\}$ that span D symbols and are present in the 2^{D-1} nodes, are needed for the optimal symbol estimation. This restriction is relaxed in the next section where more suboptimal symbol detectors are considered.

As discussed in Section III.B, a near-optimal approach is to replace $\min^*(\cdot)$ by $\min(\cdot)$ for all comparisons in (31) except for the final one (i.e. the one outside the square brackets in (31)). The complexity of the near-optimal SPS detection algorithm is given in the second column of Table 1. The exact comparison is a function of the specific design approach, the technology, and the comparison criterion used. Again, the computations required to obtain $(v_k - \sum_{j=0}^L f_j \hat{I}_{k-j})^2$, common to all algorithms considered, are not included in Table 1. Also, to perform each lookup operation, a $\delta_{i,j}^k$ (17) which is the difference between some y_i^k and y_j^k , in (28a), must be obtained. In practice, a lookup operation always follows a comparison, as given in (19b), and thus may be combined with it. First, $\delta_{i,j}^k$ may be obtained, then, its magnitude is used in the table lookup operation and its sign bit is used to select the result of comparison. Also, the time needed to perform the

'select' operation is not counted explicitly since it occurs in parallel with the look-up operation.

Nevertheless, considering that a multiplication is a few times more complex than an addition, a binary comparison, or a look-up operation, and that the complexity of an exponentiation is yet several times that of a multiplication, the substantial reduction in overall complexity is apparent. To this, one can add the other additional benefits: The (serious) problems associated with a large dynamic range, such as overflow or underflow, are (practically) avoided by replacing the summation of exponentials by values comparable to their exponents; and, finally, as stated earlier, the exploitation of the parallelism inherent in the algorithm has resulted in the derivation of a parallel structure, suitable for VLSI implementation, with a speed-up factor of N compared to a serial implementation.

IV. MORE SUBOPTIMAL DETECTORS

Simulation results show that the SPS detection algorithm derived in Section III is quite robust to uncertainty in our knowledge of the noise variance $N_0/2$ and, generally, it is relatively insensitive to the size of the look-up table (18) especially at moderate to high values of SNR. The significance of the look-up factor as a function of SNR (18) is illustrated in Figure 5 for a few values of $\delta_{i,j}^k$ (17). From (18),

$$\phi_{i,j}^k = \phi_{i,j}^k(\delta_{i,j}^k, SNR)$$

Consequently, the results of Figure 5 are independent of the channel characteristics. It can be seen that, first of all, only small values of $\delta_{i,j}^k$ (say less than $2N_0$), at relatively large quantization intervals (say 16 levels for the entire range), need

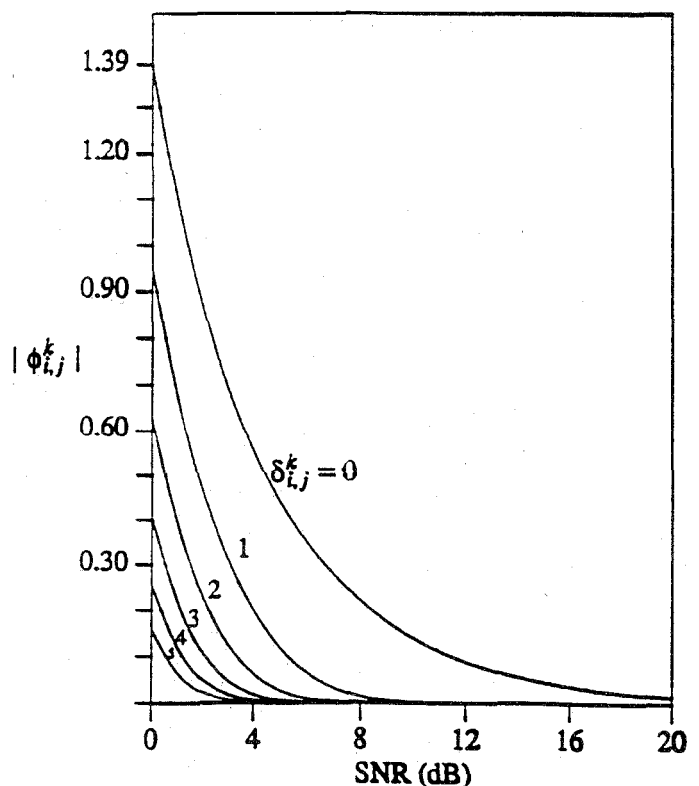


Fig. 5. Size of the look-up factor vs. SNR.

to be tabulated (hence a small look-up table) and, secondly, the look-up factor becomes negligible for moderate to high values of SNR.

This can lead to the derivation of suboptimal detectors. One such detector, referred to as Suboptimal Detector 'A1', avoids the look-up operation (i.e. ignores $\{\phi_{i,j}^k\}$) in the recursion (28) and reduces the processing required at each node to add-compare-select (ACS) only. Another one, Suboptimal Detector 'B', avoids the look-up operation both in the recursion (28) and in the symbol estimation (20),(31). In other words, each function $\min^*(\cdot)$ is replaced by the function $\min(\cdot)$.

To derive a simple algorithm for Suboptimal Detector 'B', we proceed as follows. Using (20), replacing $\min^*(\cdot)$ by $\min(\cdot)$ in (31), and interchanging the arguments of the comparison, we obtain

$$\begin{aligned} \tilde{I}_{k-D} &= \arg \left\{ \min_{l \rightarrow} \left[\min(y_0^{k-1}, y_{N/2}^{k-1}) + \min(\mu_0^k, \mu_1^k), \right. \right. \\ &\quad \left. \min(y_1^{k-1}, y_{N/2+1}^{k-1}) + \min(\mu_2^k, \mu_3^k), \dots, \right. \\ &\quad \left. \min(y_{N/2-1}^{k-1}, y_{N-1}^{k-1}) + \min(\mu_{N-2}^k, \mu_{N-1}^k) \right] \\ &= \arg \left\{ \min_{l \rightarrow} \left[z_0^{k-1} + \min(\mu_0^k, \mu_1^k), z_1^{k-1} + \min(\mu_2^k, \mu_3^k), \right. \right. \\ &\quad \left. \dots, z_{N/2-1}^{k-1} + \min(\mu_{N-2}^k, \mu_{N-1}^k) \right] \\ &= \arg \left\{ \min_{l \rightarrow} \left[\min(z_0^{k-1} + \mu_0^k, z_{N/4}^{k-1} + \mu_{N/2}^k), \right. \right. \\ &\quad \left. \min(z_0^{k-1} + \mu_1^k, z_{N/4}^{k-1} + \mu_{N/2+1}^k), \dots, \right. \\ &\quad \left. \min(z_{N/4-1}^{k-1} + \mu_{N/2-1}^k, z_{N/2-1}^{k-1} + \mu_{N-1}^k) \right] \\ &= \arg \left\{ \min_{l \rightarrow} \left[z_0^k, z_1^k, \dots, z_{N/2-1}^k \right] \right\} \end{aligned} \quad (32)$$

where (28a) is used in the second line of (32).

We recall that whereas $\{y_i^k\}$, given in (24), span the D most recent information symbols, $\{z_i^k\}$, given in (28a), only span the $D-1$ most recent information symbols. However, after replacing $\min^*(\cdot)$ by $\min(\cdot)$ in (28a), it can be noted that z_i^k is the minimum of y_i^k and $y_{i+N/2}^k$ which differ only in \hat{I}_{k-D} . Therefore, every time z_i^k is computed, the argument of the comparison, \hat{I}_{k-D} , must be stored. In other words, the reduction by a factor of 2 in area (or folding of 'space'), by mapping the 2^D members of the set $\{y_0^k, y_1^k, \dots, y_{N-1}^k\}$ onto the 2^{D-1} members of the set $\{z_0^k, z_1^k, \dots, z_{N/2-1}^k\}$, is accompanied by the introduction of 2^{D-1} surviving symbol estimates for \hat{I}_{k-D} , each associated with one of $\{z_i^k\}$.

Considering (25), the same approach, given in Section III.C that led to the mapping of the set $\{y_0^k, y_1^k, \dots, y_{N-1}^k\}$ onto the set $\{z_0^k, z_1^k, \dots, z_{N/2-1}^k\}$ may be applied repeatedly, for a total of $D-L$ times, to obtain a set $\{q_0^k, q_1^k, \dots, q_{2^L-1}^k\}$ with only 2^L components. Each mapping introduces a surviving symbol associated with every member in the set and the $D-L$ mappings introduce 2^L sur-

viving sequences, each of length $D-L$. Therefore, the algorithm for Suboptimal Detector 'B', obtained by applying the mapping used to derive (28) and (32) $D-L$ times, is as follows.

$$q_i^k = \min_{l \rightarrow} (q_{[i/2]}^{k-1} + \mu_i^k, q_{[i/2]+2^{L-1}}^{k-1} + \mu_{i+2^L}^k), \quad (33a)$$

$$i=0, 1, \dots, 2^L-1; \quad k > L+1$$

$$q_i^{L+1} = \min_{l \rightarrow} \left(\sum_{k=1}^{L+1} \mu_i^k, \sum_{k=1}^{L+1} \mu_{i+2^k}^k \right) \quad (33b)$$

$$\tilde{I}_{k-D} = \arg \left\{ \min_{l \rightarrow} (q_0^k, q_1^k, \dots, q_{2^L-1}^k) \right\} \quad (33c)$$

Equation (33) can be represented by the DG derived in Section III.A (Figure 2(b)) with 2^L nodes, where each node computes (33a). Not only is q_i^k computed, but the surviving \hat{I}_{k-L} is also stored at each node. In (33c), \tilde{I}_{k-D} is obtained from the surviving sequence which corresponds to the node with $\min(q_0^k, q_1^k, \dots, q_{2^L-1}^k)$. That this algorithm, though derived for the case $D > L$, is also valid for $D=L$ may be verified by replacing $\min^*(\cdot)$ by $\min(\cdot)$ in (19),(21).

A hybrid of the optimal SPS detector and Suboptimal Detector 'B', referred to as Suboptimal Detector 'A2', may also be derived. This can be obtained directly from the optimal SPS detection algorithm (20),(28),(31) through an approach similar to the one used to derive the 'B' algorithm. The only difference is that $\min^*(\cdot)$ and $\min(\cdot)$ are taken to be equal in the symbol estimation (20),(31) but not in the recursion (28). The result is an algorithm which is the same as (33), except that $\min(\cdot)$ is replaced by $\min^*(\cdot)$ in (33a) and (33b). The complexities of Suboptimal Detectors 'A2' and 'B' are displayed in Table 1.

The insignificance of the look-up factor with increasing SNR suggests that the performance of these suboptimal detectors should approach optimal performance as SNR increases. Figure 6 verifies this for 'B', for the specific channel and the delay constraint indicated. The curve labeled 'Threshold Detector' is obtained by examining the sign of the received symbol v_k only (ignoring the channel memory) for symbol estimation:

$$\tilde{I}_{k(\text{Thresh.Det.})} = \text{sgn}(v_k)$$

where $\text{sgn}(x)$ is -1 for $x < 0$ and $+1$ for $x \geq 0$. Furthermore, the lower-bound curve labeled 'No ISI' (for $\hat{I}_k = \pm 1$), obtained by assuming that the additive white Gaussian noise channel is not time-dispersive, represents $P_e = \{ \text{erfc}(1/\sqrt{N_0}) \} / 2$. The accuracy of the simulation results throughout this work has been such that the standard deviation for each point is less than ten percent of the mean. Moreover, simulations with a variety of channels indicated that a table size of (as low as) 16 values was adequate (i.e. only 16 values of $|\delta_{i,j}^k|$, at uniform steps, in the range $(0, 2N_0)$ were tabulated).

V. COMPARISON WITH THE VITERBI DETECTOR

The Viterbi algorithm (VA) is an efficient computational method for implementing MLSE. Let $I_J = (I_1, I_2, \dots, I_J)$ be

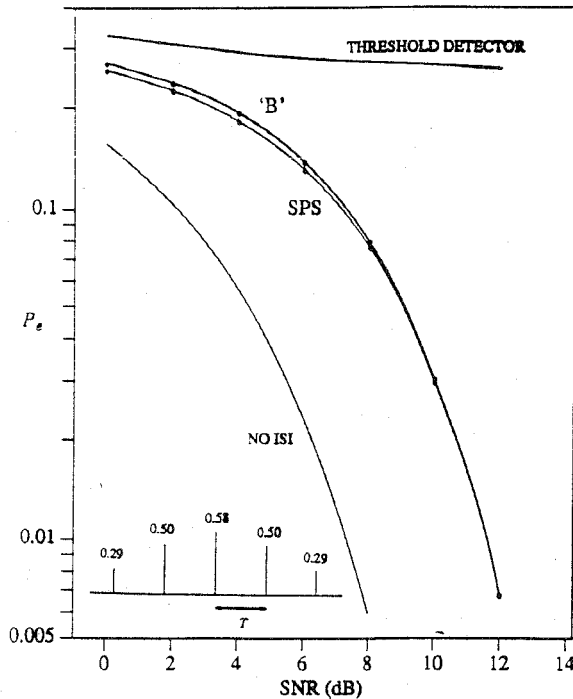


Fig. 6. Probability of a symbol error vs. SNR ($D=L=4$).

the sequence of transmitted information symbols and $\mathbf{V}_J = (v_1, v_2, \dots, v_J)$ be the sequence of the received samples (at the output of the receiver's whitening filter). Following [7, p.395], MLSE of the sequence \mathbf{I}_J is based on the maximization of the joint-probability density function

$$p(\mathbf{V}_J | \mathbf{I}_J) = p(v_J, \dots, v_1 | I_J, \dots, I_1) \\ = \prod_{k=1}^J p(v_k | I_k, I_{k-1}, \dots, I_1) \quad (34)$$

where the marginal densities on the right hand side of (34) are considered to be independent for additive white Gaussian noise.

If the length of the channel dispersion (the number of ISI terms) is L (finite), a recursive algorithm to estimate the information symbols based on MLSE can be used as follows.

Upon reception of the sample v_k , the metrics

$$w_k(\hat{I}_1, \dots, \hat{I}_{L+k}) = \min_{I_{k-L}} \left[w_{k-1}(\hat{I}_1, \dots, \hat{I}_{k-1}) + (v_k - \sum_{i=0}^L f_i \hat{I}_{k-i})^2 \right] \quad (35a)$$

are computed. The second term in the square brackets, in (35a), is the *branch metric* and the sum the *path metric*. For a binary alphabet, (35a) involves computation of 2^{L+1} path metrics and selection of 2^L surviving path metrics. Also, the \hat{I}_{k-L} suggested by each surviving path metric is stored, hence the storage of 2^L surviving sequences. At each stage, a decision will be made on the set $(\hat{I}_1, \dots, \hat{I}_m)$, $1 \leq m \leq k$, if all the 2^L surviving sequences that terminate in the symbol I_k agree on its value. Otherwise, the decision is deferred. The initial values, $\{w_{L+1}(\cdot)\}$, are given by

$$w_1(\hat{I}_1, \dots, \hat{I}_{L+1}) = \min_{I_1} \left[\sum_{k=1}^{L+1} (v_k - \sum_{i=0}^L f_i \hat{I}_{k-i})^2 \right] \quad (35b)$$

In practice, the variable delay involved in symbol detection is replaced by a fixed delay constraint through truncating the length of the surviving sequences to D most recent symbols. If D is sufficiently large, there is a high probability that all the surviving 'paths' at time k agree up to time $k-D$ [8,26]; if they do not agree, a decision (according to some strategy) is forced on the symbol I_{k-D} . It is reported that the loss in performance, due to this suboptimal decision procedure, is negligible if $D \geq 5L$ [7]. In practice, values of D in the range ($5L$ to $10L$), depending on the values of SNR, are used. Furthermore, in the VA, there are different strategies to 'force' a decision, in case the surviving sequences at time k do not agree up to time $k-D$ [8]. A few of these strategies are the 'majority decision' rule (choosing the symbol suggested by the majority of the surviving sequences), the 'minimum metric' rule (choosing the symbol suggested by the path with minimum metric), and the 'arbitrary selection' rule (choosing the symbol suggested by an arbitrary path).

The recursion in (35) can be represented by the Viterbi trellis [26] which has the same structure as the SPS detector's DG (Figure 2(b)). The Viterbi trellis may also be considered as its DG [19] and may be used directly for its VLSI implementation [14]. It consists of 2^L nodes where the processing at each node is the add-compare-select (ACS) given in (35). Based on this representation, (35) may be rewritten as

$$q_i^k = \min_{I_{i-L}} (q_{\lfloor i/2 \rfloor}^{k-1} + \mu_i^k, q_{\lfloor i/2 \rfloor + 2^{L-1}}^{k-1} + \mu_{i+2^L}^k), \\ i = 0, 1, \dots, 2^L - 1; k > L + 1 \quad (36a)$$

$$q_i^{L+1} = \min_{I_i} \left(\sum_{k=1}^{L+1} \mu_i^k, \sum_{k=1}^{L+1} \mu_{i+2^L}^k \right) \quad (36b)$$

The processing required at each node of the Viterbi trellis, given by (36a) and (36b) is shown in Figure 7. The estimate of the information symbol I_{k-D} , according to minimum-metric VA, is then given by

$$\tilde{I}_{k-D} = \arg \{ \min_{I_{i-D}} (q_0^k, q_1^k, \dots, q_{2^L-1}^k) \} \quad (36c)$$

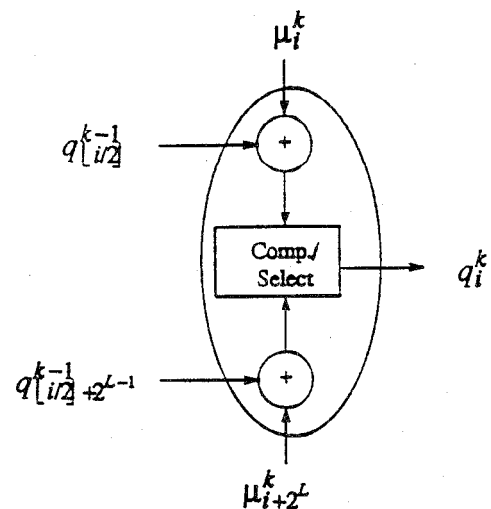


Fig. 7. Processing required at each node of Viterbi trellis.

The fixed-delay symbol-by-symbol detection algorithm and the MLSE are optimum according to two *different* criteria. The fixed-delay symbol-by-symbol detector is optimum in the sense of minimizing the probability of a *symbol* error given a delay constraint. On the other hand, MLSE is optimum in the sense of minimizing the error probability of the entire *sequence*. Nevertheless, even though VA is a method to implement MLSE, the implementation is a symbol-by-symbol process. This can be seen in (36) for minimum-metric VA. Hence, it is fair to compare the (practical) VA and the SPS detection algorithm based on *symbol* error probability, which is also the measure of performance most commonly used in data communications.

The fixed-delay symbol-by-symbol detection algorithm is, by definition of its optimality, expected to yield a lower probability of a bit error P_e compared to the Viterbi algorithm (VA) for the same delay constraint. As pointed out by Hayes [27], "optimum sequence detection considers all erroneous sequences to be equally bad" and therefore, at low SNR, "errors may lead to detected sequences that are far from the true sequence". The simulation results show that the performance of the SPS detector is indeed slightly superior to that of the Viterbi detector at low values of SNR but the two are comparable at moderate to high values of SNR. This could also be predicted from an important result shown in this work: From the comparison of (36) and (33) it is readily apparent that the minimum-metric VA and Suboptimal 'B' algorithm (derived from the optimal SPS detection algorithm) are one and the same. This result could be obtained directly through approximating the summation of exponentials in the original algorithm (7)-(9) by the largest exponential and, then, applying the transformations discussed in Section IV that led to the derivation of (34). Ungerboeck used this approximation in a different approach [6] and pointed out that, by doing so, "the 'single bit' MAP concept leading to the optimal nonlinear equalizer has been replaced by a 'sequence' ML concept. This ML solution is asymptotically optimal for high SNR."

Figure 8 shows that the gap between the SPS and the Viterbi detectors, at low SNR, remains even if the latter uses a much longer delay. An interesting range to consider is the values of the delay constraint, D , comparable to the length of the channel dispersion, L . Figure 9 shows that only the Viterbi detector with the 'minimum metric' strategy comes close to the SPS (at least for $D \leq 6L$), hence a motivation for comparing their structures.

As discussed earlier, the VA and the SPS detection algorithm have the same DG (Figure 2(b)). From the comparison of (19) and (36), or equivalently Figures 4 and 7 it can be seen that, except for an additive look-up term, the same processing is performed at every node of the two detectors. There are, 2^L nodes in the Viterbi trellis but $2^{\max(L, D-1)}$ nodes in the SPS detector's DG. This adds to the complexity of the latter as D increases beyond L . On the other hand, in the Viterbi detector, symbol estimation requires the storage and control of 2^L surviving sequences, each of length $D-L$. In a VLSI implementation, this takes a considerable area, perhaps as much as one-third of the chip area. The decision on the information sym-

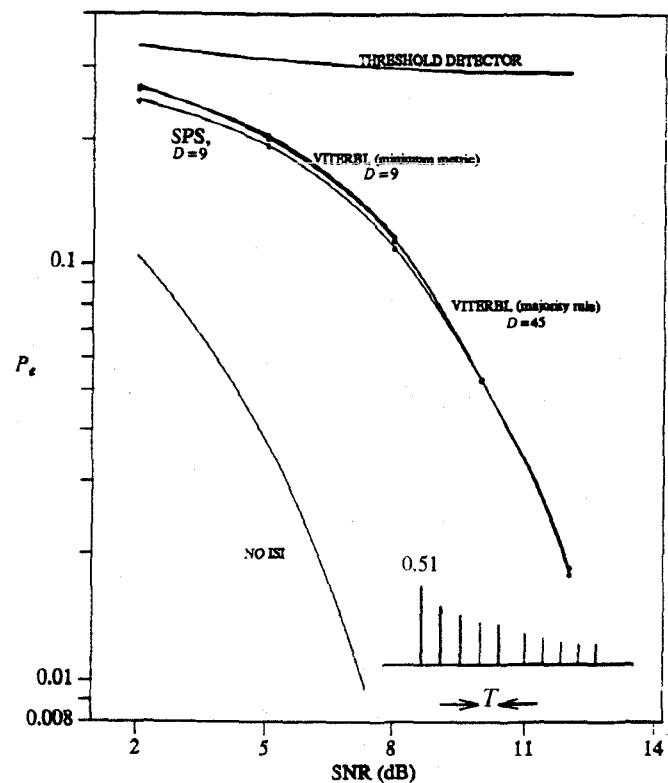


Fig. 8. Probability of a symbol error vs. SNR.

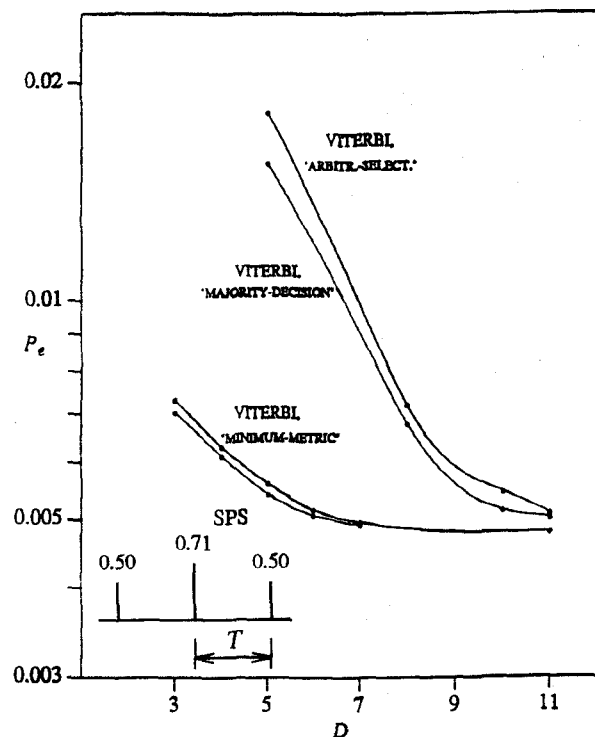


Fig. 9. Probability of a symbol error vs. delay, D , at SNR = 10 dB.

bol, made by the SPS detector, is based on the comparison of the nodes' intermediate outputs (20) at each stage, and the storage of surviving sequences is not required. In other words, for $D > L$, the Viterbi trellis operates on the L recent symbols only, and the remainder of the path history is stored

and processed outside the trellis, whereas the parallel array structure of the SPS detector operates on all the D most recent symbols (19) (or the $D - 1$ most recent symbols (28)) at once.

VI. DISCUSSION AND CONCLUSIONS

This study has shown that a practical realization of fixed-delay symbol-by-symbol detection for noisy and time-dispersive channels is possible. The mapping of the algorithm onto a fully-parallel array structure and the subsequent systematic simplifications introduce a simplified parallel symbol (SPS) detector which is several times faster and simpler than that suggested by the original algorithm.

We have shown a hierarchy of symbol detectors. The optimal SPS detection algorithm (19)-(22) (or (20), (28), and (31) for $D > L$) can be approximated by the near-optimal SPS detection algorithm (19), (20), and (23). Two suboptimal symbol detectors 'A1' and 'A2' (both discussed in Section IV) are, in turn, derived from the SPS detector. Further approximation leads to the derivation of Suboptimal Detector 'B'(33) which is shown to be the same as the minimum-metric Viterbi detector (36). The latter is superior in performance to the VA with 'majority decision' rule or 'arbitrary selection' rule with respect to minimizing the symbol error probability. The Viterbi detector with 'arbitrary selection' rule (commonly used) is, itself, an approximation to the minimum-metric Viterbi (or 'B') detector. This hierarchy from the optimal SPS detection algorithm to the 'arbitrary-selection' VA introduces a tradeoff in performance and complexity. The derivation of the optimal SPS detector together with the related suboptimal design considerations and trade-off possibilities among a number of efficient symbol detectors (including Viterbi detectors) form the main results of this work.

Our brief comparison of the SPS detector developed in this paper with the Viterbi detector shows that the former achieves a slightly better performance at low SNR's and the latter is simpler in complexity (particularly) at higher SNR's (for which large values of D are needed); otherwise, the two are comparable in complexity and performance. The simplicity of the SPS detector may make it possible to carry out a more detailed analytical comparison of its performance with respect to the Viterbi detector for specific applications and also throw light on *symbol* error performances of detectors based on *sequence* estimation.

Since the topology and the type of operations involved in the SPS detector are similar to those of the Viterbi detector, the same approaches that use the Viterbi trellis for its VLSI implementation can be applied here. The simplifications already known for VA to avoid multiplications in the branch metric computations [27] are also applicable to SPS detection algorithm. Furthermore, by modifying the expression for the branch metrics, an SPS *decoder* may be derived. A complete analysis of such designs merits further work. Finally, some of the improvements suggested for receivers that include the Viterbi detector can also be applied to the SPS detector. For example, channel truncation and estimation may be applied before symbol estimation.

APPENDIX A

This appendix provides the proof of two theorems related to the function $\min^*(\cdot)$ given in (22).

Theorem 1 For any given values A , B , and C ,

$$\min^*(A+C, B+C) = \min^*(A, B) + C$$

Proof -

$$\begin{aligned} \min^*(A+C, B+C) &= \min(A+C, B+C) - N_0 \ln(1 + e^{-|(A+C)-(B+C)|/N_0}) \\ &= \min(A, B) + C - N_0 \ln(1 + e^{-|A-B|/N_0}) \\ &= \min^*(A, B) + C \end{aligned}$$

Theorem 2 For any given values A , B , C , and D ,

$$\min^*(A, B, C, D) = \min^*[\min^*(A, B), \min^*(C, D)]$$

Proof - If $\alpha = \min(A, B, C, D)$, then according to (22),

$$\begin{aligned} \min^*(A, B, C, D) &= \alpha - N_0 \ln(e^{(\alpha-A)/N_0} + e^{(\alpha-B)/N_0} + e^{(\alpha-C)/N_0} + e^{(\alpha-D)/N_0}) \\ &= \alpha - N_0 \ln \left[e^{\max(\alpha-A, \alpha-B)/N_0} (1 + e^{-|A-B|/N_0}) + \right. \\ &\quad \left. e^{\max(\alpha-C, \alpha-D)/N_0} (1 + e^{-|C-D|/N_0}) \right] \\ &= \alpha - N_0 \ln \left[e^{\alpha/N_0} (e^{-\min(A, B)/N_0} + \ln(1 + e^{-|A-B|/N_0})) + \right. \\ &\quad \left. e^{-\min(C, D)/N_0} + \ln(1 + e^{-|C-D|/N_0}) \right] \\ &= -N_0 \ln \left[e^{-\min^*(A, B)/N_0} + e^{-\min^*(C, D)/N_0} \right] \\ &= -N_0 \ln \left[e^{-\min(\min^*(A, B), \min^*(C, D))/N_0} (1 + \right. \\ &\quad \left. e^{-|\min^*(A, B) - \min^*(C, D)|/N_0}) \right] \\ &= \min \left[\min^*(A, B), \min^*(C, D) \right] \\ &\quad - N_0 \ln(1 + e^{-|\min^*(A, B) - \min^*(C, D)|/N_0}) \\ &= \min^* \left[\min^*(A, B), \min^*(C, D) \right] \end{aligned}$$

ACKNOWLEDGEMENT

The authors wish to thank Dr. Frank Kschischang for his helpful suggestions and enlightening discussions.

REFERENCES

- [1] S.U.Qureshi, "Adaptive Equalization," *Proc. IEEE* vol.73, pp.1349-1387 (Sep.1985).

- [2] R.W.Chang and J.C.Hancock, "On Receiver Structures for Channels Having Memory," *IEEE Trans. Inform. Theory* IT-12, pp.463-468 (Oct. 1966).
- [3] K.Abend and B.D.Fritchman, "Statistical Detection for Communication Channels with Intersymbol Interference," *Proc. IEEE* vol.58, pp.779-785 (May 1970).
- [4] K.Abend, T.J.Harley, Jr., B.D.Fritchman, and C.Gumacos, "On Optimum Receivers for Channels Having Memory," *IEEE Trans. Inform. Theory* IT-14, pp.819-820 (Nov. 1968).
- [5] J.F.Hayes, T.M.Cover, and J.B.Riera, "Optimal Sequence Detection and Optimal Symbol-by-Symbol Detection: Similar Algorithms," *IEEE Trans. Commun. COM-30*, pp.152-157 (Jan. 1982).
- [6] G.Ungerboeck, "Nonlinear Equalization of Binary Signals in Gaussian Noise," *IEEE Trans. Commun. Technol. COM-19*, pp.1128-1137 (Dec. 1971).
- [7] J.G.Proakis, *Digital Communications*, McGraw-Hill (N.Y., 1983).
- [8] G.D.Forney Jr., "Maximum-Likelihood Sequence Estimation of Digital Sequence in the Presence of Intersymbol Interference," *IEEE Trans. Inform. Theory* IT-18, pp.363-378 (May 1972).
- [9] P.L.McAdam, *MAP-Bit Decoding of Convolutional Codes*, Ph.D. dissertation, Univ.of Southern California (Calif., 1974).
- [10] A.J.Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inform. Theory* IT-13, pp.260-269 (Apr. 1967).
- [11] J.K.Omura, "Optimal Receiver Design for Convolutional Codes and Channels with Memory via Control Theoretical Concepts," *Inform. Sci.* vol.3, pp.243-266 (July 1971).
- [12] H.Kobayashi, "Correlative Level Coding and Maximum-Likelihood Decoding," *IEEE Trans. Inform. Theory* IT-17, pp.586-594 (Sept. 1971).
- [13] M.V.Eyuboglu and S.U.H.Qureshi, "Reduced-State Sequence Estimation with Set Partitioning and Decision Feedback," *IEEE Trans. Commun. COM-36*, pp.13-20 (Jan. 1988).
- [14] P.G.Gulak and T.Kailath, "Locally Connected VLSI Architectures for the Viterbi Algorithm," *IEEE J. Selected Areas Commun. SAC-6*, pp.527-537 (Apr. 1988).
- [15] H.K.Thaper and J.M.Cioffi, "A Block Processing Method for Designing High-Speed Viterbi Detectors," *Proceeding ICC, Boston* (June 1989).
- [16] H.Lin and D.G.Messerschmitt, "Algorithms and Architectures for Concurrent Viterbi Decoding," *Proceeding ICC, Boston* (June 1989).
- [17] G.Fettweis and H.Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS Bottleneck," *IEEE Trans. Commun. COM-37*, pp.785-790 (Aug. 1989).
- [18] S.J.Simmons, "Breadth-First Trellis Decoding with Adaptive Effort," *IEEE Trans. Commun. COM-38*, pp.3-12 (Jan. 1990).
- [19] S.Y.Kung, *VLSI Array Processors*, Prentice Hall (N.J., 1988).
- [20] J.A.Erfanian, S.Pasupathy, and G.Gulak, "Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels," *Proc. GLOBECOM 90, San Diego* (December 1990).
- [21] J.A.Erfanian and S.Pasupathy, "Low-Complexity Parallel Structure Symbol-By-Symbol Detection for ISI Channels," *IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, Victoria, B.C., Canada* (June 1989).
- [22] J.A.Erfanian and S.Pasupathy, "Design of a Simplified Parallel Symbol Detector for ISI Channels," *Canadian Conference on Electrical & Computer Engineering, Montreal, Quebec, Canada* (Sep. 1989).
- [23] J.A.Erfanian, "Low-Complexity Parallel Structures for Symbol-By-Symbol Detection," M.A.Sc. Thesis, Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada (Jan. 1989).
- [24] S.K.Rao and T.Kailath, "Regular Iterative Algorithms and their Implementation on Processor Arrays," *Proc. IEEE* vol.76, pp.259-269 (Mar. 1988).
- [25] R.Lidl and H.Niederreiter, *Finite Fields*, Addison Wesley (1983).
- [26] G.D.Forney, Jr., "The Viterbi Algorithm," *Proc. IEEE* vol.61, pp.268-278 (Mar 1973).
- [27] J.F.Hayes, "The Viterbi Algorithm Applied to Digital Data Transmission," *IEEE Commun. Magazine* vol-13, pp.15-20 (March 1975).

Javan Erfanian received the B.Sc. degree (1986) and the M.A.Sc. degree (1989) in electrical engineering from the University of Calgary and the University of Toronto, respectively. He is a candidate for the PhD degree at the University of Toronto. His research interests include reliable transmission over channels with linear or nonlinear models, with particular attention to detection algorithms and VLSI implementable receiver structures.

He is a member of the academic staff in the department of electrical and computer engineering at the University of Toronto, acting as the academic coordinator and a lecturer for the Network Engineering Program, a strategic training program offered jointly by Northern Telecom Canada and the University of Toronto. He is teaching both of the two courses offered by Continuing Engineering Education at the University of Toronto on analog and digital communication systems. He is also serving as the Vice Chair of the Toronto Chapter of the IEEE Communications Society.

Javan Erfanian has acted as a consultant to the telecommunications industry including Northern Telecom Canada and Stentor Resource Center Inc. He was the program manager for the 1993 International Conference on Universal Personal Communications (ICUPC'93, Ottawa) and is the program manager for the 1995 IEEE Intelligent Networks Workshop (IN'95, Ottawa).

Subbarayan Pasupathy was born in Madras, Tamilnadu, India, on September 21, 1940. He received the B.E. degree in telecommunications from the University of Madras in 1963, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, in 1966, and the M.Phil. and Ph.D. degree in engineering and applied science from Yale University in 1970 and 1972, respectively.

He joined the faculty of the University of Toronto in 1973 and became a Professor of Electrical Engineering in 1983. He has served as the Chairman of the Communications Group and as the Associate Chairman of the Department of Electrical Engineering at the University of Toronto. His research interests lie in the areas of communication theory, digital communications, and statistical signal processing. He is a registered Professional Engineer in the province of Ontario. During 1982-1989 he was an Editor for (it Data Communications and Modulation) for the IEEE TRANSACTIONS ON COMMUNICATIONS. He has also served as a Technical Associate Editor for the IEEE COMMUNICATIONS MAGAZINE (1979- 1982) and as an Associate Editor for the (it Canadian Electrical Engineering Journal)(1980-1983). Since 1984, he has been writing a regular column entitled "Light Traffic" for the IEEE COMMUNICATIONS MAGAZINE. He was elected as a Fellow of the IEEE in 1991 "for contributions to bandwidth efficient coding and modulation schemes in digital communication."

Glenn Gulak received the PhD degree from the University of Manitoba.

From 1985 to 1988 he was a Research Associate in the Information Systems Laboratory at Stanford University. He is now an Associate Professor in the Department of Electrical and Computer Engineering at the University of Toronto. His research interests are in the areas of circuits, algorithms, and VLSI architectures for digital communications and signal processing applications.

Dr. Gulak has served on the ISSCC program committee (signal processing subcommittee) since 1990. He has twice received the Computer Science Students Union Undergraduate Teaching Award for courses he has taught on computer architecture at the University of Toronto.