

# REDUCED COMPLEXITY SYMBOL DETECTORS WITH PARALLEL STRUCTURES

JAVAN A. ERFANIAN, SUBBARAYAN PASUPATHY, AND GLENN GULAK

Department of Electrical Engineering  
University of Toronto  
Toronto, Ontario, Canada M5S 1A4

**Abstract** — The problem of practical realization of the optimal fixed-delay symbol-by-symbol detection algorithm is investigated. A simplified parallel symbol (SPS) detector is derived which is several times faster and simpler than that suggested by the original algorithm. A number of approximations are applied to the SPS detector which lead to the derivation of suboptimal detectors. An interesting suboptimal detector so derived turns out to be identical to the minimum-metric Viterbi detector. A comparison of the SPS and the Viterbi detectors shows that the former has a slightly better performance at low values of SNR and the latter performs a smaller number of computations (particularly) at high values of the delay constraint; otherwise, the two 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 sustained activity over the last two decades to develop improved methods of equalization. Since error probability is the most important performance measure in such applications, it is of interest to develop practical receivers suitable for VLSI implementation which optimize some measures directly related to probability of error. Two important criteria in this regard are: minimum probability of symbol error and maximum likelihood *sequence* estimation (MLSE).

Abend and Fritchman [1] derived the optimum fixed-delay 'symbol-by-symbol' detector, which is optimum in the sense of minimizing the symbol error probability given a delay constraint,  $D$  (i.e. given all the previous as well as  $D$  succeeding received samples). Nevertheless, neither this nor the related modifications or simplifications to the optimal symbol detector have received attention as being practical procedures. This is largely due to the computational burden of the algorithm, which, in particular, involves summation of exponentials and a large number of multiplications. An approach that has received tremendous attention is the Viterbi algorithm (VA), which was originally introduced for maximum-likelihood decoding of convolutional codes and was then applied to signaling on ISI channels.

This work was supported in part by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

While the structural aspects of MLSE (or the related VA) have received some attention, the criterion based on the minimum probability of symbol error has 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. For this, the practical realization of the optimal fixed-delay symbol detection algorithm derived in [1] is investigated [2,3]. The algorithm is mapped onto a fully-parallel structure suitable for VLSI implementation. Then, through systematic reformulations of the algorithm, a number of simplifications are introduced that avoid the computation of exponentials and reduce (or possibly eliminate) the number of multiplications to be performed at the expense of introducing a comparable number of simple operations of addition, comparison, and table lookup. The result is a simplified parallel symbol (SPS) detector. In Section IV, a number of suboptimal design considerations are discussed and three suboptimal symbol detectors are introduced. In Section V, the SPS detector is compared to the Viterbi detector. It turns out that one of the suboptimal detectors derived from the optimal symbol detection algorithm is identical to the minimum-metric Viterbi detector which is also a suboptimal *symbol* detector but derived from MLSE.

## II. OPTIMAL SYMBOL-BY-SYMBOL DETECTION

As in [4], let a digital transmission system with pulse amplitude modulation (PAM) be represented by an equivalent discrete-time transversal filter with tap coefficients  $\{f_k\}$ . 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 a sequence of information symbols drawn from an  $M$ -symbol alphabet ( $I_k = \pm 1$  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  $N_0/2$ .

We consider the optimal symbol-by-symbol detection algorithm under a fixed delay constraint, developed by Abend and Fritchman [1] and as presented in [4]. Let

$v_1, v_2, \dots, v_{k+D}$  be the observed received sequence, where  $D \geq L$ .<sup>1</sup> The MAP estimate of the  $k$ th information symbol, denoted  $\tilde{I}_k$ , is given by

$$\tilde{I}_k = \arg \left\{ \max_{I_k} \sum_{I_{k+D}} \cdots \sum_{I_{k+1}} U_k(I_{k+D}, \dots, I_k) \right\} \quad (2)$$

For memoryless equi-probable input symbols, the function  $U_k(\cdot)$  is obtained recursively from

$$U_k(I_{k+D}, \dots, I_k) = \exp \left[ -\left( v_{k+D} - \sum_{j=0}^L f_j I_{k+D-j} \right)^2 / N_0 \right] \times \sum_{I_{k-1}} U_{k-1}(I_{k+D-1}, \dots, I_{k-1}) \quad (3)$$

The initial value for the recursion in (3) is obtained from

$$U_1(I_{D+1}, \dots, I_1) = \exp \left\{ \sum_{k=1}^{D+1} \left[ -\left( v_k - \sum_{j=0}^L f_j I_{k-j} \right)^2 / N_0 \right] \right\} \quad (4)$$

The forbidding computational burden of this algorithm is apparent from (2)-(4). The procedure involved is particularly complicated due to the large number of the slow and complex 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 leads to overflow (or underflow) problems.

### III. SIMPLIFIED PARALLEL SYMBOL DETECTOR

To exploit the regularity and parallelism inherent in the algorithm, we follow a procedure towards a design that is suitable for VLSI implementation [2]. For simplicity, we assume  $M=2$  (binary signaling) and equi-probable and statistically independent information symbols. The results may be generalized to any alphabet size.

To simplify the notations in (3), let

$$X_j^k = U_{k-D}(\hat{I}_{k-D}^{(j)}, \hat{I}_{k-D+1}^{(j)}, \dots, \hat{I}_k^{(j)}) \quad (5)$$

and

$$\mu_j^k = \left( v_k - \sum_{i=0}^L f_i \hat{I}_{k-i}^{(j)} \right)^2 \quad (6)$$

In (5) and (6),  $j$  is the decimal value corresponding to the binary representation of a specific path. There are  $2^{D+1}$  such paths given by all possible values of the set  $\{\hat{I}_{k-D}^{(j)}, \hat{I}_{k-D+1}^{(j)}, \dots, \hat{I}_k^{(j)}\}$ . Then, the algorithm defined by (2)-(4) can be rewritten as follows.

$$\tilde{I}_k = \arg \left\{ \max_{I_k} \left[ \sum_{j=0}^{N-1} X_j^{k+D}, \sum_{j=N}^{2N-1} X_j^{k+D} \right] \right\} \quad (7a)$$

$$X_j^k = Y_{[j/2]}^{k-1} \times e^{-\mu_j^k / N_0}, \quad j=0, 1, \dots, 2N-1 \quad (7b)$$

$$X_j^{D+1} = \exp \left\{ \sum_{k=1}^{D+1} \left[ -\mu_j^k / N_0 \right] \right\} \quad (7c)$$

where

$$Y_i^k = X_i^k + X_{i+N}^k, \quad i=0, 1, \dots, N-1 \quad (7d)$$

<sup>1</sup> The algorithm for the case  $D < L$  is similar to the case  $D \geq L$  except that it considers  $L$  (rather than  $D$ ) most recent symbols for the recursion.

and  $N \equiv 2^D$ . The floor function  $\lfloor \alpha \rfloor$  denotes the greatest integer not greater than  $\alpha$ .

If one node in an index space  $(j, k)$  is allocated to the computation of each  $Y_i^k$  (Eqn.(7d)) and the dependencies between nodes in each discrete time step are represented by arrows, a *dependence graph* can be derived for the optimal symbol detection algorithm as shown in Figure 1 for  $D=2$ . The resulting structure (defined by the derived DG) has a topology similar to that of the Viterbi trellis but with *different* processing performed at each node. This is further discussed in Section V. Equation (7a), which gives the optimal estimate of the  $k$ th information symbol, uses the intermediate node outputs  $\{X_j^k\}$  given by (7b). Since  $\{X_j^{k+D}\}$ , for all  $j$ , are computed in parallel, no storage of path history is required.

The next step is to reduce the large computational burden of the algorithm. The processing required at each node is shown in Figure 2. Parallel processing does increase the throughput of the system by a factor of  $N$ , but still, the computations required at each node are complex and slow.

Let  $x_i^k = -N_0 \ln X_i^k$  and  $y_i^k = -N_0 \ln Y_i^k$ . Then, using the concept of *Jacobi's logarithm* [5], Eqn.(7d) can be rewritten as

$$e^{-y_i^k / N_0} = \exp \left\{ \left[ \min(x_i^k, x_{i+N}^k) - N_0 \ln(1 + e^{-|\delta_{i,i+N}^k| / N_0}) \right] / (-N_0) \right\} \quad (8)$$

where

$$\delta_{i,j}^k = x_i^k - x_j^k. \quad (9)$$

Let us use the following shorthand notation

$$\phi_{i,j}^k = -N_0 \ln(1 + e^{-|\delta_{i,i+N}^k| / N_0}) \quad (10)$$

Then, using (10), (8) implies

$$y_i^k = \min^*(x_i^k, x_{i+N}^k); \quad i=0, 1, \dots, N-1; \quad k > D+1 \quad (11a)$$

where

$$\min^*(x_i^k, x_j^k) = \min(x_i^k, x_j^k) + \phi_{i,j}^k \quad (11b)$$

Similarly, (7b) and (7c) imply

$$x_i^k = y_{[i/2]}^{k-1} + \mu_i^k, \quad i=0, 1, \dots, 2N-1; \quad k > D+1 \quad (11c)$$

$$x_i^{D+1} = \sum_{k=1}^{D+1} \mu_i^k, \quad i=0, 1, \dots, 2N-1 \quad (11d)$$

Note, from (10) and (11), that the knowledge of the noise variance,  $N_0/2$ , is now used in the computation of  $\phi_{i,j}^k$  only. Also, from (10),  $-N_0 \ln 2 \leq \phi_{i,j}^k < 0$  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 (11) 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  $|\delta_{i,j}^k|$ . Figure 3 shows the resulting simplification in the processing required at each node. It can be shown that further simplification is possible ( $2^{D-1}$  nodes instead of  $2^D$  nodes) for  $D > L$  by virtue of the fact that  $\mu_i^k$  and  $\mu_{i+N}^k$ , which are being added in the  $i$ th node, will be the same.

Similarly, using the same substitution in (7a) and applying the concept of *Jacobi's logarithm*, the MAP estimate of  $I_{k-D}$ , in the new structure, is obtained from

$$\tilde{I}_{k-D} = \arg \left\{ \min_{I_{k-D}} \left[ \gamma_k(0), \gamma_k(1) \right] \right\} \quad (12)$$

where

$$\gamma_k(0) = \min^*(x_0^k, x_1^k, \dots, x_{N-1}^k) \quad (13a)$$

and

$$\gamma_k(1) = \min^*(x_N^k, x_{N+1}^k, \dots, x_{2N-1}^k) \quad (13b)$$

In Eqn.(13),

$$\min^*(a_0, a_1, \dots, a_{N-1}) \equiv \min(a_0, a_1, \dots, a_{N-1}) - N_0 \ln(1 + e^{-\Delta_1/N_0} + \dots + e^{-\Delta_{N-1}/N_0}) \quad (14)$$

where  $\{\Delta_i\}$ ,  $i=1, 2, \dots, N-1$ , correspond to the difference between the minimum and every other value amongst  $\{a_0, a_1, \dots, a_{N-1}\}$ . As  $\Delta_i$  increases,  $e^{-\Delta_i/N_0}$  in (14) decreases and becomes negligible compared to '1'. Consequently, a near-optimal approach could be to keep the smallest  $\Delta_i$  (corresponding to the difference between the minimum and the next minimum) 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 to the outer binary comparison only. This way, the same look-up table used for the recursion may be used for output generation (symbol estimation). In this case, (13) may be approximated by

$$\gamma_k(0) \approx \min^* \left[ \min(x_0^k, x_1^k, \dots, x_{N/2-1}^k), \min(x_{N/2}^k, x_{N/2+1}^k, \dots, x_{N-1}^k) \right] \quad (15a)$$

and

$$\gamma_k(1) = \min^* \left[ \min(x_N^k, x_{N+1}^k, \dots, x_{3N/2-1}^k), \min(x_{3N/2}^k, x_{3N/2+1}^k, \dots, x_{2N-1}^k) \right] \quad (15b)$$

Note (by comparing Figures 2 and 3) 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 ((12) replaces (7a)). The result is a simplified parallel symbol (SPS) detection algorithm (11)-(15) 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, and that the storage of surviving sequences is not required. To this, one must add the other achieved improvements: The 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. This algorithm is optimal if (13) is used and near-optimal if (15) is used in symbol estimation given by (12).

#### IV. SUBOPTIMAL DETECTORS

Simulation results show that the SPS detector derived in Section III is quite robust against an error in our knowledge of the noise variance  $N_0/2$  and, generally, it is relatively insensitive to the size of the look-up table (Eqn.(10)) especially at moderate to high values of SNR. It can be shown that, first of all, only small values of  $\delta_{i,j}^k$  (say less than  $2N_0$ ), at relatively large quantization intervals, need 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 (11) and reduces the processing required at each node to add-compare-select (ACS) only. Suboptimal detector 'A2' keeps the additive lookup term (10) in the recursion (11) but ignores it in the symbol estimation

(12). This leads to a substantial simplification in the symbol detector by allowing its DG to collapse from having  $2^D$  nodes to  $2^L$  nodes for  $D > L$  at the expense of introducing memory for 'survivor' sequences. Finally, suboptimal detector 'B', avoids the look-up operation both in the recursion (11) and in the symbol estimation (12). In other words, each function  $\min^*(\cdot)$  is replaced by the function  $\min(\cdot)$ . It performs ACS at each node and makes a decision on  $I_{k-D}$  based on the 'dominant' value of  $\{x_i^k\}$  (Eqn.(12)) at every stage. Detector 'B' is even simpler than 'A2' in that the operations required at each of its nodes is reduced to ACS. 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 4 verifies this for the specific channel and the delay constraint indicated.

#### V. COMPARISON WITH THE VITERBI DETECTOR

Using the discrete-time model of an ISI channel given in Section II, the recursive equation for the VA is as follows:

$$w_k(I_1, \dots, I_{L+k}) = \min_{I_k} \left[ w_{k-1}(I_1, \dots, I_{L+k-1}) + (v_{L+k} - \sum_{i=0}^L f_i I_{L+k-i})^2 \right] \quad (16)$$

This involves computation of  $2^{L+1}$  probabilities and selection of  $2^L$  surviving sequences (for a binary alphabet). At each stage, a decision will be made on the set  $\{I_1, \dots, I_m\}$ ,  $1 \leq m \leq k$ , if all the  $2^L$  surviving sequences that terminate in the symbol  $I_{L+k}$  agree on its value. Otherwise, the decision is deferred.

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. 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  $k-D$  [6]. 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 node with minimum metric), and the 'arbitrary selection' rule.

The recursion (16) can be represented by the Viterbi trellis which has the same structure as the SPS detector's DG (Figure 1) but with  $2^L$  nodes (for a binary alphabet) where the processing required at each node is ACS. Based on this representation, (16) may be rewritten as

$$y_i^k = \min_{I_{k-L}} (x_i^k, x_{i+2^L}) ; \quad i=0, 1, \dots, 2^L-1 \quad (17a)$$

where

$$x_i^k = y_{[i/2]}^{k-1} + \mu_i^k, \quad i=0, 1, \dots, 2^{L+1}-1 \quad (17b)$$

and  $\mu_i^k$  is given in (6). The symbol estimation for minimum-metric Viterbi detector is then given by

$$\tilde{I}_k = \arg \left\{ \min_{I_k} (y_0^{k+D}, y_1^{k+D}, \dots, y_{2^L-1}^{k+D}) \right\} \quad (17c)$$

While SPS and Viterbi detectors are derived from different optimality criteria, they are both *symbol* detectors working on the same limited information and exhibit similarity both in performance and structure. The fixed-delay symbol-by-symbol detection algorithm is, by definition, expected to yield a lower probability of a bit error  $P_e$  compared to the Viterbi algorithm (VA) for the same delay constraint. The simulation results show that the performance (based on the symbol error probability) of the SPS detector is 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. An interesting range to consider is the values of the delay constraint,  $D$ , comparable to the length of the channel dispersion,  $L$ . The results show that only the Viterbi detector with the 'minimum-metric' strategy comes close to the SPS detector (at least for  $D \leq 5L$ ).

Equations (11) and (17) show that, except for an additive look-up term, the same processing is performed at every node of the SPS and the Viterbi detectors. Furthermore, the two detectors have the same dependencies between nodes. Thus, the similarity between their structures becomes apparent. Though derived from different criteria, suboptimal detector 'B' and minimum-metric Viterbi detector turn out to be identical.

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  $L+1$ . On the other hand, in the Viterbi detector, symbol estimation requires the storage of  $2^L$  surviving sequences, each of length several times that of the channel dispersion,  $L$ . In a VLSI implementation, this may occupy a significant portion of the total chip area. The decision on the information symbols, made by the SPS detector, is based on the comparison of the nodes' intermediate outputs (see Eqn.(13)) at each stage, and the storage of surviving sequences is not required.

## VI. DISCUSSION AND CONCLUSION

A practical realization of the fixed-delay symbol-by-symbol detection for noisy and time-dispersive channels is achieved. 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. This derivation together with the related suboptimal simplifications form the main results of this work.

Our 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 (Figure 4, where 'B' is the same as minimum-metric Viterbi detector) and the latter is simpler in complexity for high values of the delay constraint; otherwise, the two are comparable in complexity and performance. The simplicity of the SPS detector makes it possible to carry out a more detailed comparison of its performance with respect to the Viterbi detector for specific applications.

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 [7] can be applied here. The simplifications, already known for VA, to avoid multiplications in the branch metric computations [8] are also applicable to SPS detection algorithm. Furthermore, by modifying the expression for the branch metrics, SPS decoder may be derived. A complete analysis of such designs merits further work.

## ACKNOWLEDGEMENTS

The authors wish to thank Frank Kschischang for his helpful discussions and comments.

## REFERENCES

- [1] K.Abend and B.D.Fritchman, "Statistical Detection for Communication Channels with Intersymbol Interference," *Proc. IEEE* vol.58, pp.779-785 (May 1970).
- [2] 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).
- [3] J.A.Erfanian, J.Dao, G.Gulak, and S.Pasupathy, "Realization of the SPS Detection Algorithm on a Parallel VLSI Architecture," 15th Biennial Symposium on Communications, Kingston, Ontario, Canada (June 1990).
- [4] J.G.Proakis, *Digital Communications*, McGraw-Hill (N.Y., 1983).
- [5] R.Lidl and H.Niederreiter, *Finite Fields*, Addison Wesley (1983).
- [6] 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).
- [7] 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).
- [8] J.F.Hayes, "The Viterbi Algorithm Applied to Digital Data Transmission," *IEEE Commun. Magazine* vol-13, pp.15-20 (March 1975).

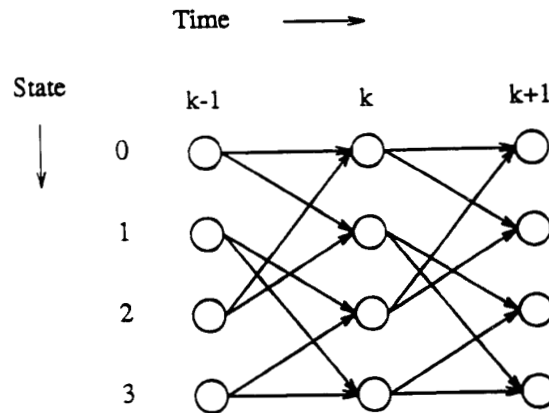


Fig 1 — The dependence graph of the symbol detector.

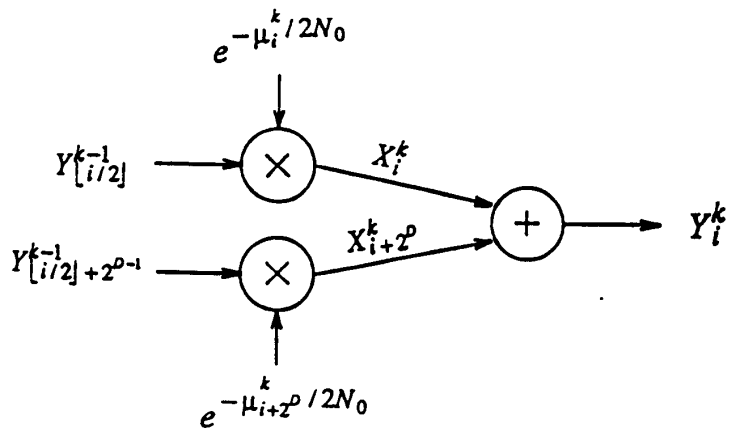


Fig 2— The processing required at each node before simplification.

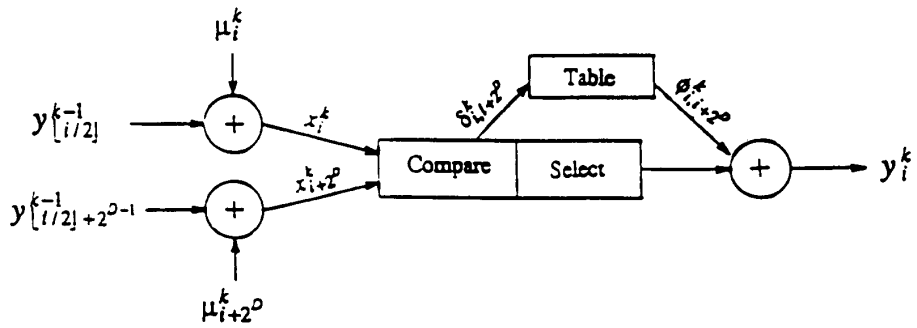


Fig 3— The processing required at each node after simplification.

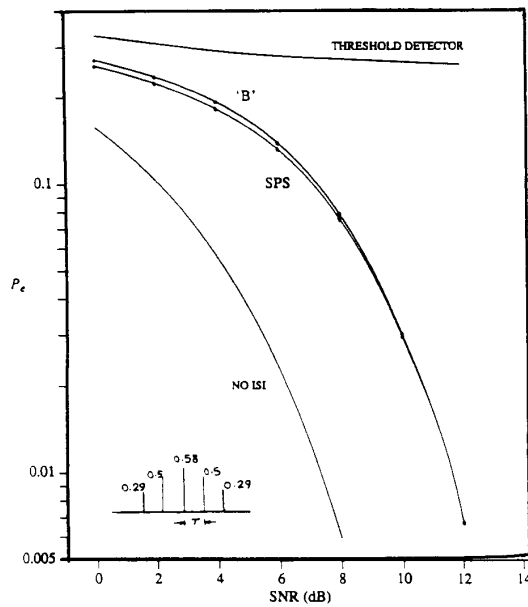


Fig 4— Probability of a symbol error vs. SNR;  $D=L=4$ .