

Realization of the SPS Detection Algorithm on a Parallel VLSI Architecture

JAVAN ERFANIAN, JOSEPH DAO, GLENN GULAK, AND SUBBARAYAN PASUPATHY

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

Abstract — Recently, the simplified parallel symbol (SPS) detection algorithm, which is an efficient approach to implement the fixed-delay symbol-by-symbol detection on ISI channels, has been developed. In this paper, the realization of this algorithm suitable for a fully parallel VLSI architecture is studied and the corresponding area/data-rate tradeoff is presented.

SUMMARY

The fixed-delay optimal symbol-by-symbol detection algorithm, as developed by Abend and Fritchman [1], is of interest by virtue of the fact that an important performance criterion in most communication systems is the symbol error probability. Through simplification of this algorithm and exploitation of its parallelism, a simplified parallel symbol (SPS) detection algorithm was derived [2,3]. A dependence graph (DG), which shows the flow of computations in time, was derived for the SPS detector. This can be restructured as shown in Figure 1 for a binary alphabet with the delay constraint $D = 2$. In this representation, the breadth of the DG includes 2^D nodes (states). The result is similar in structure to the Viterbi trellis. Thus, we are motivated by the possibility of implementing the SPS detector using known architectures [4] for Viterbi algorithm (VA, which is an efficient method of implementing *maximum likelihood sequence estimation*). As in [5], let the received sequence v_k be 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 (binary alphabet is considered in here), L is the length of the channel dispersion, $\{f_0, \dots, f_L\}$ is a set of discrete channel coefficients, and $\{\eta_k\}$ are independent identically distributed Gaussian random variables with zero mean and variance $N_0/2$. The algorithm estimates I_{k-D} given the observed received sequence v_1, v_2, \dots, v_k , where the delay constraint $D \geq L$. The recursive equations, derived in [2] but based on the DG of Figure 1, are now given by

$$y_i^k = \min(x_i^k, x_{i+N}^k) + \phi_{i,i+N}^k; \quad i=0, 1, \dots, N-1; \quad k > D+1 \quad (2a)$$

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

The research was supported by grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada and Information Technology Research Center (ITRC).

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

where $N=2^D$. In (2), $\phi_{i,j}^k = -N_0 \ln(1 + e^{-|\delta_{i,j}^k|/2N_0})$ and is tabulated explicitly as a function of $\delta_{i,j}^k = x_i^k - x_j^k$. Since the delay constraint is given to be D , a decision \tilde{I}_{k-D} on the information symbol I_{k-D} can be made based on the received sample v_k . As discussed in [2], a near-optimal procedure requires choosing the two minimum values among $\{x_0^k, x_1^k, \dots, x_{N-1}^k\}$, denoted by x_{u1}^k and x_{u2}^k , and the two minimum values among $\{x_N^k, x_{N+1}^k, \dots, x_{2N-1}^k\}$, denoted by x_{l1}^k and x_{l2}^k . Then,

$$\tilde{I}_{k-D} = \arg \left\{ \min_{I_{k-D}} \left[\min(x_{u1}^k, x_{u2}^k) + \phi_{u1,u2}^k, \min(x_{l1}^k, x_{l2}^k) + \phi_{l1,l2}^k \right] \right\} \quad (3)$$

In the computation of the recursion defined by (2), the following term, which we will refer to as the *branch metric* (as in VA), is to be evaluated.

$$\mu_i^k = (v_k - C_i)^2 = v_k^2 + C_i^2 - 2C_i v_k \quad (4)$$

where $C_i = \sum_{j=0}^L f_{k-j} \hat{I}_j$ and the values of $\{\hat{I}_{k-L}, \hat{I}_{k-L+1}, \dots, \hat{I}_k\}$ correspond to the states $\lfloor i/2 \rfloor$ at time $k-1$ and $\{i \bmod N\}$ at time k . The energy term v_k^2 is independent of the state and can be ignored. Then, substituting for C_i and using an approach similar to that in [6] the new branch metrics may be obtained as follows.

$$\lambda_i^k = -\frac{1}{2} \sum_{n=0}^L \sum_{j=0}^L \hat{I}_n \hat{I}_j r_{n-j} + \sum_{j=0}^L \hat{I}_j z_j^k \quad (5)$$

where $r_{n-j} = f_{k-n} f_{k-j}$ (the discrete representation of the channel autocorrelation coefficients) and $z_j^k = v_k f_{k-j}$ (the sample output of a filter matched to the discrete channel).

The first term on the right-hand side of (5) does not depend on v_k and can be precomputed. For a binary alphabet, the $\{I_j\}$ in the second term take on the values ± 1 and, consequently, no multiplication is required in the computation of branch metrics.

The recursive expressions given by (2) (and represented by Figure 1 for $D=2$) involve a series of add-compare-select (ACS) operations on intermediate data values. The Viterbi detector has a trellis similar in topology to Figure 1 and it (too) involves ACS operations. Thus, as mentioned earlier, one is motivated to explore the possibility of implementing the SPS detector using known VLSI architectures for the VA. In order to illustrate the similarities that can be exploited in this regard, we restrict attention to a parallel implementation as provided by a shuffle-exchange embedding. However, we should also note the main differences between the two algorithms:

- (i) The SPS detection algorithm requires an additive integer constant obtained from a small look-up table following each ACS operation.
- (ii) For a channel dispersion of length L and a binary alphabet, the VA requires 2^L states or path metrics. In contrast, $2^{\max(D,L)}$ states are required for the SPS detector with a delay constraint of D . Note, however, that the number of branch metrics that need to be computed, at each stage, is 2^{L+1} for

both detectors.

(iii) The storage of 2^L survivor sequences is required in VA. Survivor sequences are not required in the SPS detection algorithm.

The output generation (symbol estimation) is given by (3). A suboptimal SPS detector may be obtained by just using the minimum value from one set (corresponding to either $I_{k-D} = -1$ or $I_{k-D} = 1$) and an *arbitrary* value from the same set to perform the look-up operation. In this case we only need to find the minimum value of $\{x_0^k, x_1^k, \dots, x_{N-1}^k\}$ and the minimum value of $\{x_N^k, x_{N+1}^k, \dots, x_{2N-1}^k\}$. The problem becomes how to efficiently (minimum steps and minimum wiring to route the operands) compare N different values to find the minimum. If a shuffle exchange graph is used to layout the SPS structure, several methods can be used for output generation. One method is to find the minimum of the individual necklaces [7] of the shuffle exchange layout first, then proceed to find the minimum of these necklaces to get the final result. For an N node shuffle exchange graph, there are N/D number of necklaces and, at most, D nodes in each necklace. To compare values in a necklace, D steps are required. (It takes D steps to pass a value around a necklace with D nodes.) No extra comparators are needed for this step as the comparator used for the compare/select operation can be used again but with different inputs. A simple multiplexer will be needed to route the corresponding inputs for each part of the algorithm. To compare the minimum from all necklaces, an additional $\log(N/D)$ steps is needed. To select the minimum amongst the N/D necklaces, $(N/D) - 1$ comparators are needed. The time complexity of this method grows as $O[D \log(N/D)]$ which is equal to $O[(\log N)^2 - (\log N)(\log(\log N))]$.

A time optimal method of finding the minimum of a set of N numbers is to use a comparison tree. The time complexity grows as $O(\log N)$. If a shuffle-exchange graph is used to layout the SPS detector, a comparison tree structure can also be implemented using the same connections. The same comparators used in the recursion can be used but with different inputs. Another method which seems to arise naturally from the shuffle exchange implementation is the bitonic search. A bitonic search network can be implemented efficiently using a shuffle exchange graph. It involves a series of shuffle operations and compare-exchange operations. For an N node (a sequence of N different values) shuffle exchange graph, the bitonic search can be used to sort the N different values into an ascending order sequence. No extra comparator is needed although some simple control circuits (mainly multiplexers) are needed to control the broadcast of signals. The complexity grows as $O(\log^2 N)$. This grows more rapidly than the previous necklace-comparison method. For the extra price we pay in the time complexity, we get the added benefit that the whole sequence will be sorted at the end. This is useful when we want to implement (3) and hence the near-optimal SPS detection algorithm. Table 1 summarizes the complexity of the above methods.

The SPS detection algorithm may therefore be realized on fully-parallel VLSI architecture using simple operations of addition, comparison, and table look-up. The branch metrics may be computed using (5). The parallel array structure may be implemented, based on the DG of Figure 1, using known VLSI architectures for VA (considering the differences discussed above). Finally, output generation may be performed using one of the search methods summarized in Table 1.

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 and S.Pasupathy, "Design of a Simplified Parallel Symbol Detector for ISI Channels," *Canadian Conference on Electrical & Computer Engineering*, Montreal, Quebec, Canada (Sep. 1989).
- [4] 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).
- [5] J.G.Proakis, *Digital Communications*, McGraw-Hill (N.Y., 1983).
- [6] J.F.Hayes, "The Viterbi Algorithm Applied to Digital Data Transmission," *IEEE Commun. Magazine* vol-13, pp.15-20 (March 1975).
- [7] F.T.Leighton, *Complexity Issues in VLSI*, The MIT Press (Mass., 1983).

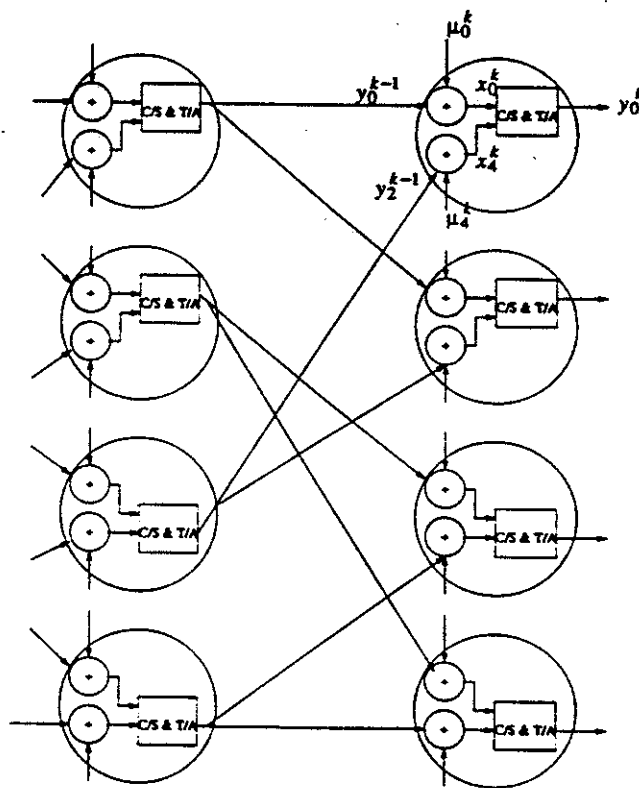


Figure 1. New DG for SPS detector after regrouping (D=2).
(C/S & T/A = compare/select + table look-up and add)

TABLE 1 Output Generation

Method	Time Complexity	Δ Area Increased
Comparison Tree	$O(\log N)$	$O(1)$
Necklace	$O[(\log N)^2 - (\log N)(\log(\log N))]$	$O(N/D)$
Bitonic Search	$O[(\log N)^2]$	$O(1)$