SIMPLIFIED PATH METRIC UPDATING IN THE M ALGORITHM FOR VLSI IMPLEMENTATION.

Luis González and Emmanuel Boutillon

Ecole Nationale Supèrieure des Télécommuncations
46, rue Barrault
75634 Paris Cedex 13, France
Luis.Gonzalez@enst.fr, emmanuel.boutillon@enst.fr

ABSTRACT

A VLSI structure for path metric updating in the M algorithm is presented. The architecture is based on the combination of a modified Batcher's odd-even merging network and a bitonic selection procedure. A feature of the trellis structure allows to replace an existing solution based on two $2M$–item sorting operations by three M–item sorting operations with an additional one-layer bitonic merge. These three sorting networks and the bitonic merging procedure permit a reduction of up to 50% in hardware complexity.

1. INTRODUCTION.

Trellis coding techniques have proved their good performance in different fields such as channel coding, source coding and sequence estimation. Conventionally, trellis coding schemes are composed of a finite state machine (FSM) and a trellis search algorithm.

The Viterbi algorithm is the optimum search algorithm for trellis encoding/decoding schemes [1]. Hardware realizations are rich and well documented [2, 3]. However, it is well known that its complexity grows exponentially with the number of states. Hence, in applications such as source coding or sequence estimation, where a large number of states is needed, the Viterbi algorithm becomes prohibitively complex. Thus, other suboptimum algorithms with lower complexity must be used.

The M algorithm has come to be a good alternative and is being widely adopted in source coding applications because it produces good performance codes while much fewer computations than the Viterbi algorithm are carried out. Contrary to the Viterbi algorithm, in the M algorithm only the best M paths are retained at every instant. This implies a direct exploration of the trellis where each iteration is composed of 3 steps:

1. extentsion: the M paths at time t are extended to their two succesors at time $t+1$. For the $2M$ path thus generated, the new path metric of the $2M$ paths are also computed.
2. supression of merged paths: it can occurs that two distinct paths merge at time $t+1$. If notihing is done, the effective number of explored paths is reduced and performance will decrease. Thus, when two paths merge, the one with the highest path metric has to be discarded.
3. selection of the best M surviving paths, i.e., the M paths having the the smallest path metrics.

Steps 2 and 3 imply the use of sorting circuits. However, the hardware complexity of sorting circuits can be very large[1].

In [4], the authors describe a multiprocessor architecture that performs the sorting operation according to two criteria. A first sorting is performed with respect to the state associated to each path so that the paths with the same state will be adjacent; this way, merged paths can be discarded during the second sorting. The second sorting is performed with respect to the path metric so that the best M paths are selected and the merged paths are discarded. The sorting is based on a modified Batcher's sorting network [5]. Note that for this solution, two sorting operations of a list of $2M$ items have to be performed. Other authors propose to delete the merging paths in an "a posteriori" way, with a delay, by detecting the paths that do not have the same bit pattern in the surving path [6, 7]. This solution allows the use of only one sorting operation during the path selection. Nevertheless, it also decreases the performance since merged paths are not immediately discarded and it requires the use of a register exchange technique which is not the most efficient technique [8].

[1]In practice, the M algorithm is advantageous only when the number of surviving paths M is much lower than the number of trellis states.

In this paper, we propose a new structure that performs two sorting operations. The first one is performed with respect to the path metrics and the second one with respect to the state numbers. Using a feature of the trellis structure that allows the sorting operation to be split into two smaller sorting procedures, the hardware complexity can be reduced up to 50% compared to [4].

2. THE M ALGORITHM.

A 2^{K-1}–state FSM can be implemented with a K–bit shift register. The $K-1$ LSBs define the current state of the FSM and the $K-1$ MSBs define the next state. A trellis diagram is used to depict the evolution in time of the different transitions between states visited by an input sequence. The parameter K is called the constraint length of the trellis. The concatenation of transitions generates different paths through the trellis. The aim of the search algorithm is to find the best path that minimizes a distortion criterion between the symbols associated to that path and the input sequence.

Figure 1 shows an example of how the M algorithm works. At every time instant t, only the best M paths are retained. Associated to each path is a value, called path metric, which is a distortion measure that indicates how well the search process is being performed. The path metric is the accumulation of transition metrics. The transition metric is the distortion introduced by the distance between the symbol associated to a trellis transition and the input symbol. The path metric is the criterion to select the best M paths. In addition to the path metric, a state number and a decision word are associated to each path. The state number indicates the trellis state visited by the surviving path at time t. The decision word indicates the trellis transition that has been selected to be appended to the surviving path. These decision words define the decoded sequence. We assume that there are 2 transitions leaving each path. Hence, the size of the state number and the decision word is $K-1$ and 1 bits respectively. Two transitions might arrive to a given state. If the surviving transition is the upper one, the decision bit is set to zero. Otherwise, the decision bit is set to one.

At the next time instant, $2M$ new paths are created together with their associated states, decision words and path metrics. The M paths having the lowest path metrics are selected. This process is repeated until all the input sequence has been processed. Then, starting from the last generated decision bit of the path having the least distortion, the decision bits of this path are traced back until the first generated decision bit. This sequence of decision bits is the decoded sequence.

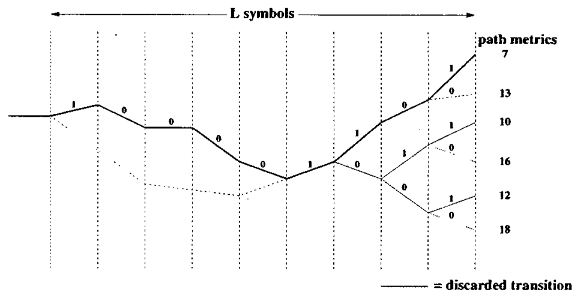

Figure 1: Operation of the M algorithm (M=3).

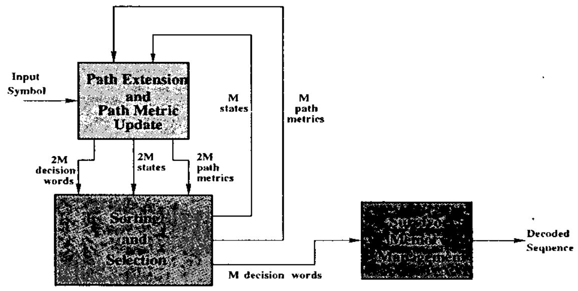

Figure 2: Proposed Architecture.

3. PROPOSED ARCHITECTURE.

A block diagram of the proposed architecture is presented in figure 2. In the path extension and path metric updating block, the new $2M$ paths are generated together with their associated states and decision words. In the sorting and selection block, the M paths having the least distortions are selected and the merged paths are discarded. After the best paths are selected, their decision words enter the surviving memory management block for sequence decoding. Only the first two blocks are treated here. The reader is referred to [8] for a description of the survivor memory management.

3.1. Path Extender and Transition Metric Computation.

Figure 3 shows the internal architecure of the path extender. Let $C(t)$ be the set of state numbers of the M paths at time t. Let $S^k(t)$ be the state number of path k at time t. We have

$$C(t) = \{S^k(t)\}_{0 \leq k < M-1} \qquad (1)$$

The state number can be represented by the binary contents of the shift register associated to the state

$$S^k(t) = (i^k_{t-1}, i^k_{t-2}, \cdots, i^k_{t-K+1})_2 \qquad (2)$$

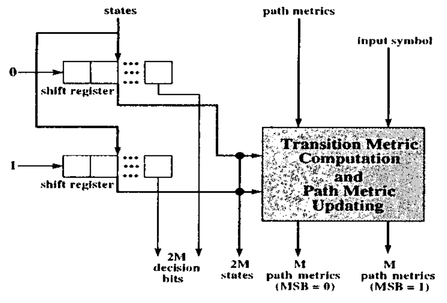

Figure 3: Architecture for Path Extension.

where the index 2 indicates a binary representation.

During the extension operation, each state $S^k(t)$ are extended to its two succesors of time $t+1$, $S^k(t+1) = (0, i^k_{t-1}, \cdots, i^k_{t-K+2})_2$ for the arrival of a 0 and $S^k(t+1) = (1, i^k_{t-1}, \cdots, i^k_{t-K+2})_2$ for the arrival of a 1. We can cluster the $2M$ extended paths in two sets $C_0(t+1)$ and $C_1(t+1)$ corresponding to the set generated by the arrival of a 0 and a 1 respectively

$$C_0(t+1) = \{(0, i^k_{t-1}, \cdots, i^k_{t-K+2})_2\}_{0 \le k < M-1} \quad (3)$$
$$C_1(t+1) = \{(1, i^k_{t-1}, \cdots, i^k_{t-K+2})_2\}_{0 \le k < M-1} \quad (4)$$

We can notice that $C_0(t) \bigcap C_1(t) = \emptyset$ since the MSB of the state number is different in the two sets. Moreover, we can notice that the order of the state numbers is maintained during the extension process from $C(t)$ to $C(t+1)$. In other words, if $S^k(t) > S^{k+1}(t)$ then $S^k_0(t+1) \ge S^{k+1}_0(t+1)$ and $S^k_1(t+1) \ge S^{k+1}_1(t+1)$. Thus, if the set $C(t)$ is in sorted order, then $C_0(t+1)$ and $C_1(t+1)$ are also in sorted order.

These two properties can be used to simplify the sorting process of the M algorithm.

3.2. First Method for Selecting the Best Paths.

As we have seen, the $2M$ paths can be split into two sets according to the MSB of the states associated to each path. A first method for selecting the best M paths is shown in figure 4. A sorting operation and the rejection of merged paths can be performed on each set. This will discard all merged paths and deliver the two sets in sorted order. Finally, a simple merging procedure will suffice to find the best M paths, not necessarily in sorted order.

Figure 5 shows the inner structure of the two sorting networks. As explained in [4], the first sorting operation is performed with respect to the state number so that the merged paths will be adjacent at the end of this first sorting. When two merged paths are detected, the one with the largest metric is discarded. Then, a sorting operation according to the path metric

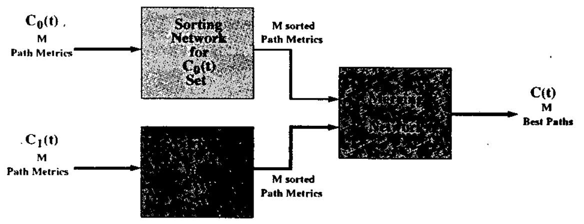

Figure 4: Block Diagram of the Sorting and Selection Operation.

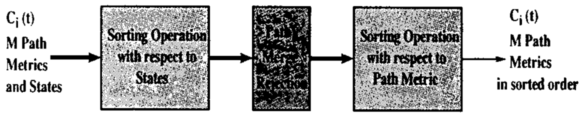
$C_i(t)$ M Path Metrics and States
Sorting Operation with respect to States
Sorting Operation with respect to Path Metric
$C_i(t)$ M Path Metrics in sorted order

Figure 5: Inner Structure of the $C_0(t)$ and $C_1(t)$ sorting networks.

is performed [2]. This way, instead of performing two $2M$–item sortings, we perform four M–item sortings and a merging procedure to select the best M paths between the two sets.

The merging network in figure 4 which finds the best M path metrics of the two sets can be easily performed with the first column of a bitonic sorter. In [5] it was shown that if the input of a bitonic sorting network is a bitonic list of $2M$ items, then, after the first M–size layer of comparison elements, the list will be divided in two M–item lists, the upper list will contain the smallest path metrics whereas the lower list will contain the largest metrics. These two lists are not necessarily in sorted order. Thus, since the first two M–item sorters of figure 4 will deliver the two sets in sorted order, we only need a column of M comparison elements to find the smallest metrics of the two sets.

3.3. Second Method for Selecting the Best Paths.

The architecture in figures 4 and 5 can be further improved. Since $C(t)$ is in sorted order, the sets $C_0(t+1)$ and $C_1(t+1)$ will be in sorted order too. Thus, the sorting operation according to the state number can be performed on $C(t+1)$, that is, after the merging procedure of the two sets. This way, when the new $2M$ paths will be generated, the sets $C_0(t+2)$ and $C_1(t+2)$ will be in sorted order and merging states will be adjacent. This procedure is shown in figure 6. The operation is started with a rejection of the merged paths. Next, the path metrics on each set are sorted and merged in order to find the best paths of the two

[2]The two sorting operations can be performed simultaneously if the sorting process of the path metric is modified.

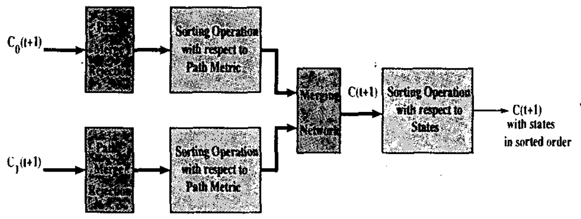
$C_0(t+1)$
Path Merge Rejection
Sorting Operation with respect to Path Metric
$C_1(t+1)$
Path Merge Rejection
Sorting Operation with respect to Path Metric
Merging Network
C(t+1)
Sorting Operation with respect to States
C(t+1) with states in sorted order

Figure 6: New Method for the Selection of the M Best Paths.

Table 1: Hardware Complexity for the Selection of the M Best Paths.

M	Mohan et al. [4]	Two Set Sorting	Hardware Reduction (%)
4	38	17	45%
8	126	61	48%
16	382	197	52%
32	1086	589	54%
64	2942	1661	56%
128	7678	4477	58%

sets. Finally, the best M paths are sorted according to the state number so that, at the next path extension, the states on each set will be in sorted order and merged paths can be discarded.

The main advantage of this architecture is the reduction in hardware. According to [4], the two sorting operations of $2M$ items each, require $2 \cdot [(\log_2^2 2M - \log_2 2M + 4)2^{log_2 2M-2} - 1]$ comparisons elements. With our procedure only $3 \cdot [(\log_2^2 M - \log_2 M + 4)2^{log_2 M-2} - 1]$ comparison elements of the three sorters and $\frac{M}{2}$ additional comparison elements of the bitonic merger are needed. Table 1 shows the number of comparison elements required by the two methods and the reduction in hardware achieved with our procedure.

4. CONCLUSIONS.

An architecture for path metric updating in the M algorithm has been presented. A new structure which combines an odd-even sorting network and a dichotomic selection procedure was proposed for the selection of the best M paths. The use of the selection procedure reduces in a great extent both the hardware required and the processing time. A new way of implementing the rejection of unmerged paths was proposed, based on the fact that unmerged paths are likely to be found only in half of the new created paths. This way, not all the $2M$ paths have to be tested.

5. REFERENCES

[1] G.D. Forney Jr. "The Viterbi Algorithm". *Proc. IEEE*, vol. 61:pp. 268–278, March 1973.

[2] G. Gulak and T. Kailath. "VLSI Architectures for the Viterbi Algorithm.". *IEEE Journal on Selected Areas in Communication.*, vol. 6:pp. 527–537, April 1988.

[3] H. Kaeslin M. Biver and C. Tommasini. "In-place updating of Path Metrics in Viterbi Decoders". *IEEE Journal of Solid-State Circuits*, vol. 24:pp. 1158–1160, August 1989.

[4] Seshadri Mohan and Arun K. Sood. "A Multiprocessor Architecture for the (M,L)-Algorithm Suitable for VLSI Implementation". *IEEE Transactions on Communications*, vol. 34:No. 12, December 1986.

[5] K. E. Batcher. "Sorting Networks and Their Applications". *in Proc. AFIPS 1968 Spring Joint Comput. Conf. Montvale, NJ*, vol. 32:307–314, 1968.

[6] Stanley J. Simmons. "A Bitonic-Sorter Based VLSI Implementation of the M-algorithm". *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages pp. 337–340, June 1989.

[7] Peter A. Bengough and Stanley J. Simmons. "Sorting-Based VLSI Architectures for the M-algorithm and T-algorithm Trellis Decoders". *IEEE Transactions on Communications*, vol. 43(2/3/4):514–522, April 1995.

[8] Emmanuel Boutillon and Luis González Pérez. "Trace Back Techniques Adapted to the Surviving Memory Management in the M Algorithm.". *submitted to ICASSP2000*, Istambul:Turkey, 2000.