TRACE BACK TECHNIQUES ADAPTED TO THE SURVIVING MEMORY MANAGEMENT IN THE M ALGORITHM.

Emmanuel Boutillon and Luis González

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

ABSTRACT

A new architecture for survivor memory management in the M algorithm is presented. So far, classical implementations of the survivor memory management employ the register exchange procedure. The architecture presented here is based on the trace back procedure used in the Viterbi Algorithm. Using a new pointer which indicates the number of the surviving path given by the sorting operation during the path metric updating operation, all the trace back techniques that have been proposed for the Viterbi algorithm can be employed for the M Algorithm. This architecture is specially attractive for large values of M and L in wich case the register exchange approach is impractical due to power consumption and to the area required for wiring. In addition, a combination of the register exchange and the trace back procedures is also presented. The combination of these algorithms reduces both the information to be stored and the processing time.

1. INTRODUCTION.

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 this paper, we focus on the survivor memory management of the M Algorithm. We present a new architecture based on the trace back algorithm. To our knowledge, no candidate architectures for the M algorithm employing the trace back procedure have been reported. All reported architectures use the register exchange approach [1, 2, 3, 4]. In this approach, the L most recent symbols of the M surviving paths are stored in a register, L being the decoding depth of the trellis. During the selection of the best M paths, that is, during the sorting operation, all L bits of each surviving path are moved and placed in the corresponding line of registers according to the sorting operation. The main advantage of this approach is that it is composed of $L \times M$ identical and simple elements (a register and a multiplexor). However, the disadvantages are the small density of storage elements, the large area required by the interconnexion network and power consumption since all registers might switch states at every clock cycle. As a consequence, register-exchange-like implementations are limited to small values of M and L. A good alternative for memory management with large values of M and L is the trace back algorithm. The main advantage of this approach is its simplicity and the large density of the storage elements. However, the required storage elements and the decoding delay are larger. In addition to the trace back architecture, we

[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.

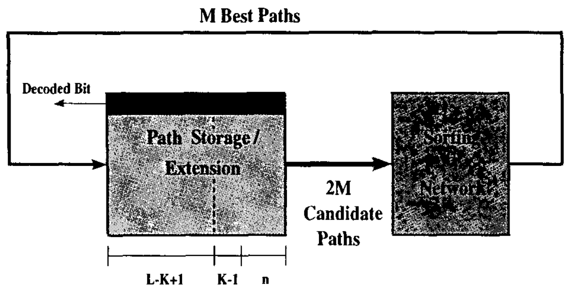

Figure 1: General Structure for Register Exchange Implementation of the M Algorithm.

also present a structure which combines both the register exchange and the trace back algorithms. This way, we benefit from the advantages of both procedures.

The reminder of the paper is organized as follows. In section 2 we present the register exchange procedure applied to the M algorithm. The new trace back structure for memory management applied to the M algorithm is given in section 3. The combination of both methods is presented in section 4. Finally, conclusions are given in section 5.

2. REGISTER EXCHANGE PROCEDURE FOR THE M ALGORITHM.

In the register exchange implementation of the M algorithm, all the bits composing both the surviving and the candidate paths are stored in a register bank, each row of registers corresponds to a given path. Each row stores the $n-$ bit path metric and the L bits that compose the trellis path. A general structure employed by most of the reported architectures is shown in figure 1. In this structure, all the bits in a row move together through the sorting network, thus, the comparison elements of this network must be able to store the bits and direct them to the right position. The path storage and extension block is an array of registers with multiplexors that support data flow such as up or down shiftings. Interconnexion networks are used to simplify this massive flow of information [1], or they are processed serially in order to obtain a more compact structure [3]. In any case, this movement of information is power consumming and the required wiring is large. However, the latency of this structure is very small. From $t = L$, the bits coming out of the register bank are the decoded sequence.

3. TRACE BACK PROCEDURE FOR THE M ALGORITHM.

In the same manner as in the Viterbi algorithm, the trace back procedure stores the decisions about which transition survives. We will assume that there are two transitions leaving each state. If the upper transition arriving to a certain state is selected, the decision bit is set to zero. Otherwise the decision bit is set to one. The path metric updating block described in [5] delivers M decision bits. These decision bits constitute a decision vector. Each time an input symbol enters the decoder, a decision vector is generated and stored in the memory. The memory address where this vector is stored is given by the number of the input symbol t. When the last decision vector is generated (at $t = L_{S_{in}}$ where $L_{S_{in}}$ is the size of the input sequence), the trace back procedure is performed. The aim is to find the sequence of visited states by tracing the decision bits back from $t = L_{S_{in}}$ until $t = 0$.

The placing of the decision bits is not fixed in the M algorithm. They are given by the sorting operation in increased order according to the path metrics (see [5]). Hence, additional information must be generated so that the trace back will be able to know where the right decision bit is located in the decision vector. In other words, we need some information indicating the place in the decision vector of address $t = t - 1$ where the next decision bit has to be read. Thus, $\log_2 M + 1$ bits are required per surviving path; $\log_2 M$ bits indicate the next $\log_2 M + 1$ bits that have to be taken from the decision vector of address $t = t - 1$. The remaining bit indicates the surviving transition. This bit serves to generate the sequence of states visited by the surviving path. The decision vector is thus composed of $M(\log_2 M + 1)$ bits.

In order to perform the trace back operation, the state corresponding to the least distortion path is taken and its decision bits in address $L_{S_{in}}$ are read. The $\log_2 M$ MSBs indicate the position, among the $M(\log_2 M + 1)$ bits of the decision vector in address $L_{S_{in}} - 1$, where the next decision bits have to be taken. This procedure is repeated until all the decision bits are read. The LSBs of the decision bits serve to generate the decoded sequence. Figure 2 shows an example of the trace back operation for a 4-state trellis and $M = 2$. Dashed lines indicate the discarded transitions. Continuous lines indicate the surviving transitions and the bold line is the surviving path. The decision memory contains 4 decision bits per address. The first two bits correponds to the least distortion path and the last two bits to the second least distortion path given by the sorting operation in the path metric updating pro-

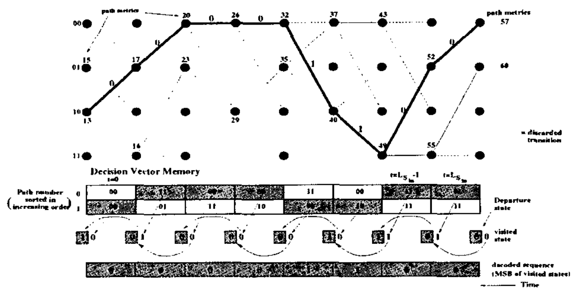
Figure 2: Example of the Trace back algorithm in the M algorithm (M=2).

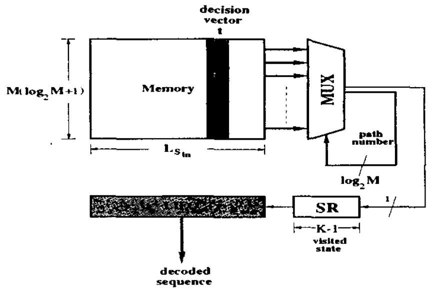

Figure 3: Trace back Architecture for the M algorithm.

cess. This 4–bit word can thus be divided in two 2–bit words. The MSB of each one of the 2–bit words indicates whether the first or the last two bits are taken from the next decision vector. The gray boxes in the decision memory indicate the decision bits that produce the decoded sequence. Finally, in order to produce the decoded sequence, the visited states are generated by shifting their value to the left and appending the LSB of the decision bits at the LSB position. This is shown by the arrows in the decision memory and the visited states row. The MSB of each visited stated is a bit of the decoded sequence. The bits in the gray boxes of the visited states are the decoded sequence.

The hardware implementation of the trace back algorithm is presented in figure 3. At each clock cycle, a decision vector is read from the memory. The multiplexor selects the $\log_2 M + 1$ bits indicated by the $\log_2 M$ bits of the previous clock cycle. The LSB of the decision bits enter the shift register in order to create the visited state. Finally, the MSB of the shift register is stored in a LIFO. The LIFO will contain the entire decoded sequence. This is a very simple implementation of the trace back algorithm. Nevertheless, all the architectures that have been proposed to implement the trace back procedure for the Viterbi algorithm can be employed for the M algorithm, namely, more elaborated architectures that take into account the decoding depth of the trellis in order to reduce the decoding delay.

4. COMBINATION OF THE REGISTER EXCHANGE AND TRACE BACK PROCEDURES.

It is possible to combine the register exchange and the trace back procedures in order to obtain a memory management structure that benefits from the advantages of both algorithms. For example, we can reduce the size of the register exchange bank of figure 1 to $2M \times (\log_2 M + m)$ bits for the surviving and candidate paths. This way, every m cycles the decision bits corresponding to the $M \cdot m$ surviving trellis transitions and the $M \cdot \log_2 M$ bits corresponding to the M state numbers at time $t = t - m$ are stored in the trace back memory. During m cycles, the decision bits that indicate the trellis transitions that compose the surviving paths are stored in the register bank together with the $\log_2 M$ bits that indicate the path number of each surviving path at time $t = t - m$. Then, after m cylces, the $M(\log_2 M + m)$ bits coming out of the sorting network which form the decision vector are stored in the trace back memory. During the trace back operation, every trace back step will decode m bits of the decoded sequence. The remaining $\log_2 M$ bits indicate the place of the surviving path m cycles before. The procedure is repeated in the same way as in section 3.

Figure 4 shows an example of the operation of this combined procedure for $M = 2$ and $m = 2$. Every $m = 2$ cyles, $M(\log_2 M + m) = 6$ bits are stored at each memory location. The three first bits correspond to the least distortion metric. From this 3 bits, the MSB indicates the position of the surviving path $m = 2$ cycles before and serves to fetch the next 3 bits either from the first or last 3 bits of the next memory location. During the trace back procedure, $m = 2$ bits are decoded at each step. Thus, in only 4 cycles the entire sequence is decoded. The arrows indicate the two bit output at each step and the location of the next 3 bits composing the surviving path. Notice that for this special case, since $m = K - 1$, every 2 cycles we can know the visited state two cycles earlier.

There are several advantages of this procedure over the single register exchange and trace back procedures. In one hand, the required storage is reduced because instead of storing $m \cdot M(\log_2 M + 1)$ bits every m cycles, with this new procedure only $M(\log_2 M + m)$ bits are stored every m cycles. There is an economy of $m \cdot M$ bits. On the other hand, the latency of the structure is also reduced. Instead of decoding one single bit every

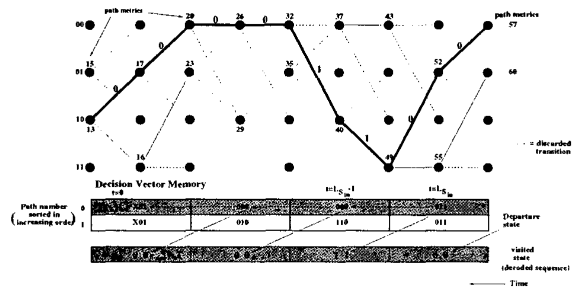

Figure 4: Example of the Combined Register Exchange and Trace Back procedures (M=2).

trace back step, m bits are decoded in a single trace back step with this method. In other words, to decode p bits with the single trace procedure, p trace back steps are needed whereas with the proposed method, only $\frac{p}{m}$ trace back steps are required. In addition to these advantages, the proposed method has a larger density than the single register exchange procedure and a smaller latency than the single trace back algorithm.

5. CONCLUSIONS.

A new architecture for survivor memory management in the M algorithm, based on the trace back procedure has been presented. The trace back architecture is more compact than the register exchange approach and simplifies or avoids the massive flow of information. This allows low power and small area circuit desings. This way, an increase in the number of surviving paths and in the decoding depth is possible so that more paths can be traced and hence the performance of the trellis encoding/decoding scheme can be improved. In addition to the trace back procedure, a system which combines the register exchange and trace back algorithms has also been presented. This way, we benefit from the advantages of both methods, namely, the compactness and low power consumption of the trace back, and the small number of memory locations and decoding delay of the register exchange. Finally, notice that the same achitectures that have been employed and proposed for the Viterbi algorithm can be used for the M algorithm if the $\log_2 M$ bits indicating the path number are considered.

6. REFERENCES

[1] 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.

[2] Stanley J. Simmons. "A Nonsorting VLSI Structure for Implementing the (M,L) Algorithm". *IEEE Journal on Selected Areas in Communications*, vol. 6(3):538–546, April 1988.

[3] 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.

[4] 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.

[5] Luis González Pérez and Emmanuel Boutillon. "Simplified Path Metric Updating in the M Algorithm for VLSI Implementation.". *submitted to ICASSP2000*, Istambul:Turkey, 2000.