# A generalized precompiling scheme for surviving path memory management in Viterbi decoders

Emmanuel BOUTILLON, Nicolas DEMASSIEUX
Telecom Paris , E.N.S.T, 46 rue Barrault, 75634 PARIS CEDEX 13, FRANCE
e-mail : BOUTILLON, DEMASSIEUX@ELEC.ENST.FR

***Abstract-*** **The management of the surviving path memory in Viterbi's algorithm is generally performed by Trace-Back or Exchange.Register. A generalized method using precompiled trace-backs is presented. Resolution by a graphical method is proposed and three examples are solved.**

## I. INTRODUCTION

Management of the surviving path memory in Viterbi's algorithm is an important problem especially when considering VLSI implementation. Two well known methods, Exchange Register (ER) and Trace-Back (TB) both have their limitations [1,2,3,4]. ER is a direct implementation allowing high data throughput, since the critical path consists in one mux and one latch. In return, when the code size increases, ER becomes quickly critical in terms of area and power dissipation. TB based on RAMs can lead to good integration density but the number of data to store is large. Other solutions using a combination of ER and TB have already been proposed by Fettweis in [6] and by Sparsø in [7], allowing important savings in area and power dissipation. In this paper, a generalization of the methods combining TB and ER is proposed : the precompiled TB. A graphical representation of the management of the surviving path memory is presented. Three interesting solutions of precomiled trace-back are discussed in terms of trade-off between the low number of bits to store for the ER and the integration density of RAMs which is made possible by the TB approach.

## II. SURVIVING PATH MEMORY

A convolutional encoder is a finite state machine which evolution is controlled by the succession of bits in the message. For each coding cycle, an information is generated by the encoder concerning the evolution of its internal state. The decoder searches for the most likely evolution of the encoder as a function of the arriving information.

The encoder is a simple shift register of length $\upsilon$, which receives the bits $u_0$, $u_1$, ... , $u_n$ to be transmitted. The $2^\upsilon$ possible states are represented by the content of the $\upsilon$ registers. For example, at time t and t+1, the encoder is respectively in the states $u_t u_{t-1}..u_{t-\upsilon+1}$ and $u_{t+1}u_t..u_{t-\upsilon+2}$. The succession of the encoder states is usually represented by a path in the transition diagram of the encoder, or trellis (fig. 1). At each decoding cycle, the decoder gets an information on the encoder transition. The most likely previous state, among

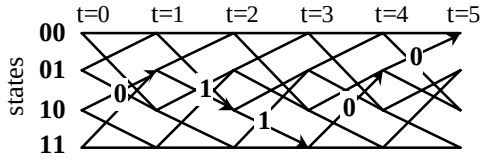


Fig. 1 : Example of trellis for $\upsilon$=2. The bold path indicates the succession of the encoder states at times 0 to 5. The message bit which changes the encoder state between t=i and t=i+1 is indicated on the corresponding branch. For example, from t=1 to 2, the state of the encoder changes from 01 to 10. This is due to the arrival of one 1 in the shift register.

the $N=2^\upsilon$ possible ones, is usually determined by the branch metric unit and by the Add-Compare-Select unit (ACS) [5]. The information is encoded in the form of a binary decision vector $V_t$ of length N. Let $b_i^t$ be the value of the $i^{th}$ component of $V_t$, and $a_t a_{t-1}...a_{t-\upsilon+1}$ the binary representation of the state i. If $b_i^t=0$, the most likely antecedent of the state i at time t-1 is $a_{t-1}...a_{t-\upsilon+1}0$; if $b_i^t=1$, it is $a_{t-1}...a_{t-\upsilon+1}1$. The bit $b_i^t$ corresponds to the last bit of the antecedent of i at time t-1. To each node of the trellis is associated a path, called surviving path, built up by the succession of the states which are selected (fig. 2). It is possible to demonstrate that all surviving paths are converging, with a high probability, toward a unique path, provided that the number of successive branches is equal or larger than the convergence length, L. This unique path is the decoded message. A theoretical description of Viterbi's algorithm can be found in [5]. Two classes of architecture allow the management of the surviving memory :

- The Exchange Register uses a direct hardware implementation of the trellis structure. A bank of L registers, containing the L previous bits of the surviving path, is associated to each node of the trellis. The surviving paths are shuffled and updated according to the $V_t$ decisions. In the example of fig. 2 the first bits of the surviving path, corresponding to the node 01, are 1010, at time t=4. This node takes the decision 0 at time t=5, and its new surviving path is the concatenation of 0 with the surviving path of its

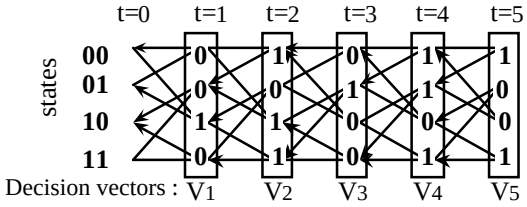


Fig. 2 : At each decoding cycle, the vector $V_t$ provides the most probable antecedents of each node of the trellis, at time t-1. A decision 0 corresponds to the upper branch, a decision 1 to the lower one.

antecedent for t=4 (0 & 0001 = 00001). The hardware implementation required by the ER involves N*L multiplexed registers. This method is limited to low values of L and N, because of the problem of area and power dissipation. Fig. 3 shows a graphical representation of the ER algorithm.

- In the second method, named trace-back (TB), one starts from a randomly chosen node and its surviving path is followed on a depth L+ H, in order to get H decoded states. It may be necessary to use several parallel TB in order to respect the throughput constraint (same frequency for decoding and generating V(t) vectors). Fig. 4 shows a classical solution proposed by Rader in [4], which achieves easily a decoding throughput of 25 Mbit/s. The main advantage of the TB is the very dense storage of the vectors in RAMs. However the number of vectors to be stored and the decoding delay are large.

## III. General Precompiled Scheme

The underlying idea of the precompiled architectures is as follows. A continuous Exchange Register carried out on the whole surviving path leads to inacceptable area and power dissipation. It is nevertheless possible to perform a partial Exchange Register by generating and then following pieces of the surviving paths of h-bit width over a depth k. We will call this scheme partial Exchange Register ER(h,k) and the generation duration of an ER(h,k), i.e. k decoding cycle, a k-macro-cycle. Every k-macro-cycle, this partial ER gives us the possibility to obtain for each node, the sequence of h bits of its surviving path between t-k+h and t-k+1. In other words, the ER(h,k) precompiled TB consists in executing every k-macro-cycle the following operation :

- from t = n.k+1 to t = n.k+h, the $V_t$ vectors are used as in a classical ER i.e they initialize the first bit of the surviving paths and control the shuffle.
- from t = n.k+h+1 to t = (n+1).k, the $V_t$ vectors are only used to shuffle the partial surviving paths of length h that have been created before.

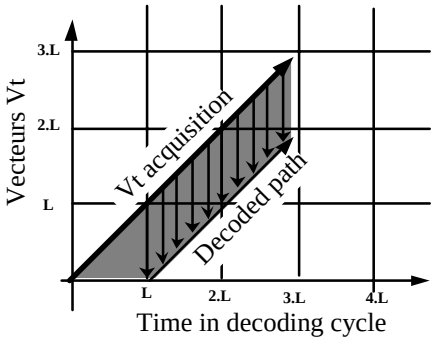


Fig. 3 : Graphic representation of the ER algorithm.
The horizontal axis represents time, and the vertical axis the $V_t$ vectors. The curve x=y ($V_t$ acquisition) indicates that $V_t$ is known only after time t. The vertical arrows indicate that, at each decoding cycle, all the surviving paths over a depth L of all the node are known. The horizontal distance between $V_{tt}$ acquisition and decoded state indicate the decoding delay L.

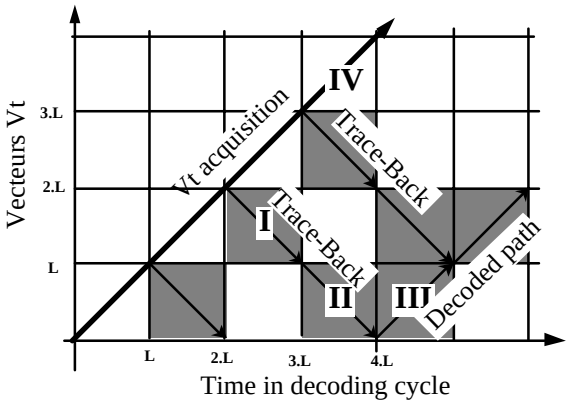


Fig. 4 : Graphical representation of a classical TB : H=L and two TB are performed for each cycle. In the graphic, the duration of a decoding cycle is equal of the duration of one trace-back..

I- Backtracking of the surviving path : The trace-back starts at t = 2.L, with the decision vectors $V_{2.L}$. In L cycles (from t=2.L to t=3.L), a surviving path is back-tracked over a depth L : at time t=3.L, we have the corresponding decoded state of $V_L$. This is depicted by the arrow I.

II- Effective decoding of data

III- Rearrangement of the message

The decoding delay is 4.L (horizontal distance between $V_t$ acquisition and decoded path). Only 3.L vectors should be stored since the vectors $V_t$, which are deleted from area II, can be replaced by vectors from area IV.

At time t= (n+1).k, the partial ER contains the surviving paths from t = n.k+h to t = n.k+1 of every node of the trellis. These surviving paths are stored in a look-up table. With the condition h•υ, this look-up table allows to know the antecedent at time n.k of each node of the trellis and enables a very fast "precompiled" TB. Fig. 5.a gives an illustration of a precompiled trace-back ER(4,7). At time t = (n+1).7, the partial exchange register has generated the surviving path of all the nodes between t=n.7+1 and t=n.7+4. With the table created (fig 5.b), one can execute the equivalent of 7 TB between time (n+1).t and n.7 during the convergence phase or obtain the 4 decoded states between t-7 and t-4 after the convergence in only one cycle of precompiled TB. The

number of vectors to be stored will be small and the area of the reduced ER becomes acceptable. We can make an interesting trade-off between the low number of bits to store for the ER and the integration density of RAMs which is made possible by the TB approach. With this formalism, the conventional solution TB and ER can be expressed in the form ER(1,1) and ER(L,L).

## IV. CASE STUDIES

### *A : ER(h,h) precompilation*

Paaske, Pedersen and Sparsø, have proposed in [7] a solution of the type ER(h,h). The TB is then executed at the speed of h states per decoding cycle (fig. 6). For example, for L=56 and h=8, it is possible by 8 precompiled TB, to restore the surviving path on a depth 64 and to decode 64-56 = 8 states. If the precompiled trace-back is executed at the same rate than the decoding, the constraint flow is respected.

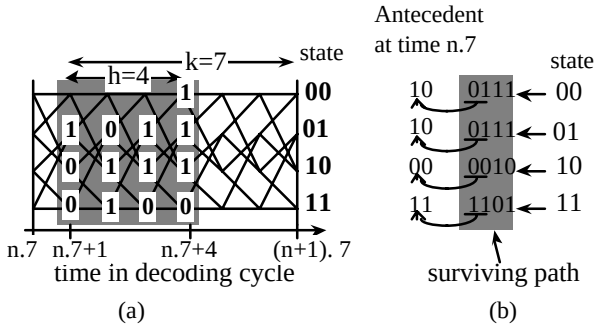


Fig. 5 : Representation of all surviving paths at the end of the computation of an ER(h,k) partial Exchange Register.
(a) Representation of the known surviving path in the trellis. (grey area).
(b) Register contents of the partial ER at the end of a k-macro cycle.

A precompiled RAM implementation can be achieved only in full-custom. In fact, in writing mode, partial exchange register results have to be stored (h vertical vectors of length N), and in reading mode, a part of the surviving path of the current node must be read (a horizontal h bits word). An efficient implementation of the precompilation unit can be found in [7].

### *B : ER(υ,L) precompilation*

This method consists in using an ER unit of width υ working in a macro-cycle of L decoding cycles. This ER(υ,L) allows the emulation of a group of υ consecutive vectors passing through a classic exchange register. This reduced exchange register unit produces a decoded state every L cycles. This state is the starting point of a TB of depth L. This method gives on one hand a decrease of the storage depth from 3L (pure TB) to 2L. On the other hand, with an adequate organization of the memory bank, one needs to read or write only once per decoding cycle (fig. 7) while the classic TB requires to read from and write to the RAM every decoding cycle. Since RAMs are generally on the critical path of the Viterbi decoder, the method is able to double the clock

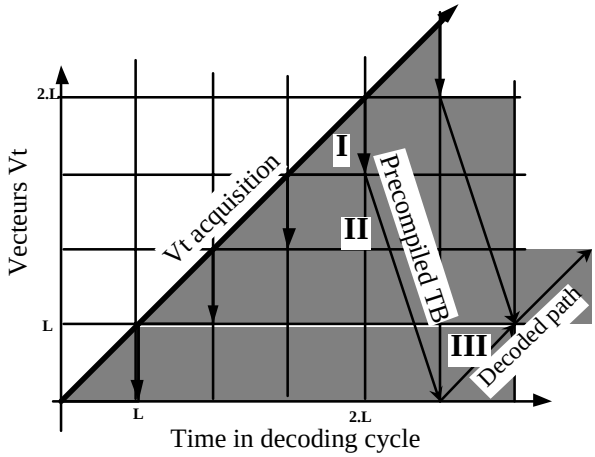


Fig. 6: Organization of the ER(h,h) precompiled, with h = L/3. We can see that the slope of the precompiled trace-back is -h.
I- Precompilation
II- Precompiled trace-back, the slope of the line is -h
III- Rearrangement of the message

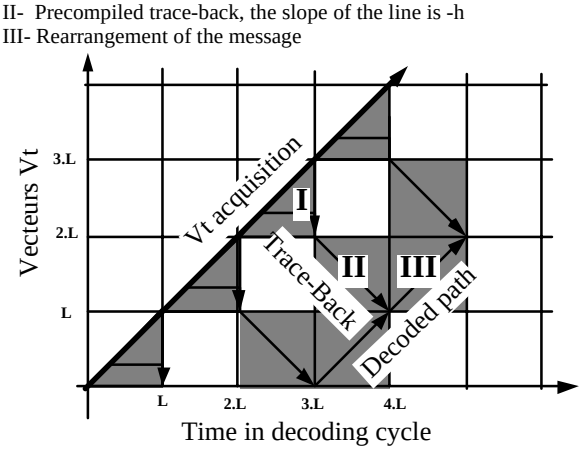


Fig. 7 : Organization of the ER(υ,L) precompiled
I- Precompilation
II- Effective decoding of data
III- Rearrangement of the message

frequency and the throughput used in a classical TB whereas the silicon area is slightly reduced.

The ER unit can have the same topology as an ACS block. Every L cycles (a macro cycle), the register are initialized with the ACS number they represent. The register contents are shuffled afterwards during L cycles, according to the ACS decisions in such a way that the register value corresponds to the number of the antecedent ACS at the beginning of the macro-cycle. After L cycles, every surviving path having converged, the registers have the same value : the one of the decoded state. Once the convergence is established, power dissipation is reduced to the clock activity, whereas it is not the case of the classical ER (fig. 8).

The ER(υ,L) method is very well suited to standard cell implementation because it does not need specific RAM but

standard ones. Ongoing implementation of this architecture for L=100 in a 0.8 µm standard-Cell technology gives an area of 12 mm$^2$, and decoding throughput of 38 Mbit/s for a maximum power dissipation of 850 mW.

*C : ER(υ,L/υ) precompilation*

This method is a precompilation of the TB operation by generating, with an ER unit, a look-up table between state at time t and the previous state of the surviving path corresponding to t = t - L/υ. This table is constructed by following a group of υ consecutive vectors through a register exchange over L/υ cycles and by recording part of the surviving path obtained. From those precompiled tables, it is

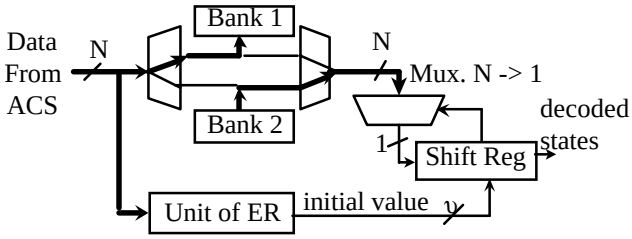


Fig. 8: Memory organization of an ER(υ,L)

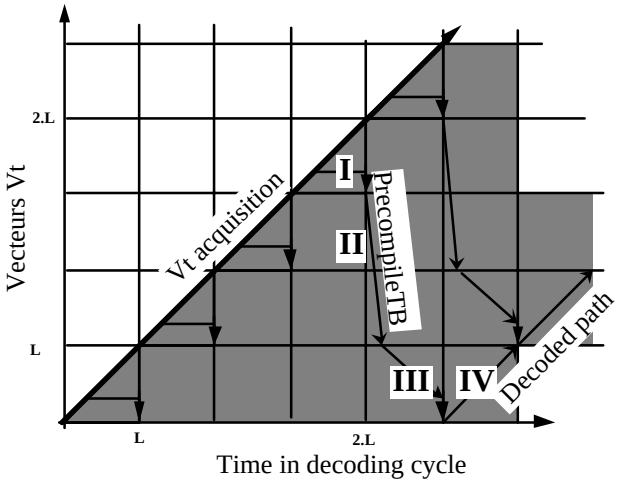


Fig. 9 : ER(υ,L/υ) precompiled, the slope of the precompiled TB is L/υ. The total number of vectors to store is L+L/υ in which υ$^2$ are precompiled type.

possible to get, after υ memory read cycles, a decoded state which will be used as a starting point for a classical TB (fig 9). The backtracking of the surviving path is very fast : hence the amount of data to be stored is small.

## V PERFORMANCES ESTIMATION

To quickly compare the silicon areas of the different solutions, we will consider that the area is essentially devoted to store the decision vectors $V_t$. We can define an "Equivalent Stored Vector" unit or ESV which corresponds to the RAM area for one vector $V_t$. The area cost of a stored vector in an ER will be R ESV, with R being a parameter dependant on technological factors and on the number of states of Viterbi trellis. Typically, R is roughly equal to 8 for a 64-state Viterbi. With this measure, the implementation area of the TB described in [1] is 3*L and the implementation area of the ER is R*L (fig. 10).

ESTIMATION AREA OF STUDIED SOLUTIONS IN ESV

| Type | ER memory | Precompiled TB | Classical TB | Delay |
|---|---|---|---|---|
| Exch. Reg. | R.L | 0 | 0 | L |
| Classical TB | 0 | 0 | 3.L | 4.L |
| ER(L,υ) | P.υ | 0 | 2.L | 3.L |
| ER(h,h) | R.h | $p^*.h^2+(p-1).h$ | 0 | $p.h^2+h$ |
| ER(L/υ,υ) | P.υ | $υ^2$ | $L.(1+υ)/υ-υ^2$ | L.(2+υ)/υ |

$p^*$ = number of precompiled TB executed after convergence, $p = \mathrm{Sup}\left(\frac{L-h}{h.(h-1)}\right)$

For a υ=6 Viterbi with L > 50, this ER(υ,L/υ) precompiled method leads to a smaller silicon area than the ER(h,h) while the same throughput is maintained. On the other hand, a hardware implementation of this structure would be more complicated, because it must necessarily include a precompiled back-tracking unit as well as a classical one. In

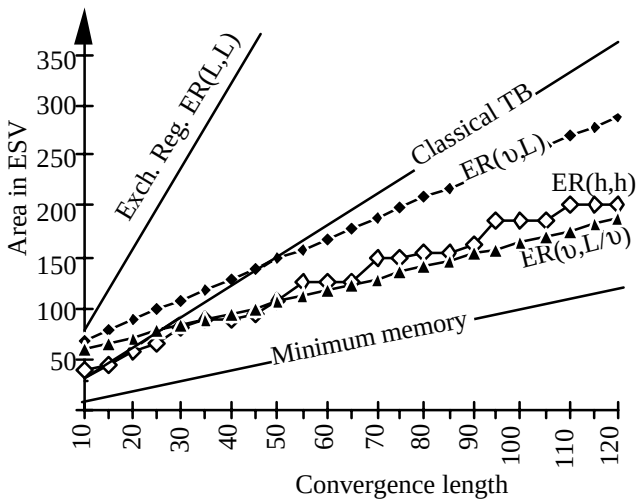


Fig. 10: Architecture area for υ=6, For the ER(h,h) curves, h is chosen as function of L to minimize the area.

terms of area, ER(υ,L) is less efficient than ER(h,h) and ER(υ,L/υ) but its implementation is straightforward by using classical standard cells. Each of these three architectures has, hence, its fields of application.

## CONCLUSION

We have presented in this paper a unification of the TB and ER algorithm. The graphical method presented here can

be easily generalized. The number of trace-back or precompiled trace-back can be arbitrary and the duration of a backtracking cycle can be different than the decoding cycle. If $\alpha$ is the ratio between the TB's cycle duration and the decoding cycle duration, graphically, the slope of a classical and precompiled back-tracking will be respectively $-\alpha$ and $-k.\alpha$. This graphical resolution method eases the search of a good solution, especialy for high speed Viterbi decoder where we can find $\alpha<1$.

## REFERENCES


[1] O. Collins, F. Pollara, "Memory management in traceback Viterbi decoders", DA Progress Report 42-99, Jet Propulsion Lab., Pasadena, CA, (Nov. 1989).

[2] R. Cypher and C. B. Shung, "Generalized traceback techniques for survivor memory management in the Viterbi algorithm", in Globecom, pp 1318- 1322, (Dec. 1990).

[3] G. Feygin, P. G. Gulak and F. Pollar, "Survivor sequence memory management in Viterbi decoder,", in Third IBM Workshop on ECC, pp 72-90, San Jose, CA, (Sept. 1989).

[4] C. M. Rader, "Memory management in a Viterbi decoder", IEEE TC, Vol. com, n°9, pp. 1399-1401, (Sept. 1991).

[5] G. D. Forney, "The Viterbi algorithm," Proc. IEEE, Vol.61, pp.218-278, (Mar. 1973).

[6] G. Fettweis, "Algebraic survivor memory management for the Viterbi detectors", to appear in conf. proceeding of IEEE Int. Conf. on Communications, ICC'92, Chicago, paper n°313.4, ( June 14-18 1992).

[7] E. Paaske, S. Pedersen, J. Sparsø, "An area-efficient path memory structure for VLSI implementation of high speed Viterbi decoders", INTEGRATION, The VLSI journal 12, pp. 79-91, (1991).