A Low Complexity Soft-Output Viterbi Decoder Architecture

Claude Berrou, Patrick Adde, Ettiboua Angui and Stéphane Faudeil

Integrated Circuits for Telecommunications Laboratory
Ecole Nationale Supérieure des Télécommunications de Bretagne, France

ABSTRACT This paper presents a means to adapt the classical architecture of a Viterbi decoder to make it able to provide soft (weighted) decisions. After a theoretical justification of the proposed method, based on Battail or Hagenauer-Hoeher algorithms, we detail the new architecture which leads to a real-time circuit, the size of which is roughly twice the size of the classical Viterbi decoder. In order to appreciate the quality of the weighting method, an application to the decoding of concatenated convolutional codes, with the proposed soft-output decoder as the inner decoder, is examined.

1. Introduction

The Viterbi algorithm [1], in its basic operation, has been devised so as to yield binary decisions and is unable to weight them by any information of confidence or reliability; indeed, this additional information is generally unnecessary when a single code is used. When concatenated coding is considered with a convolutional code as the inner code, the performance of the global decoder can be significantly improved if the inner decoder is able to provide soft or weighted decisions for the subsequent stage. In this paper, we propose a realistic means to adapt the classical Viterbi decoder architecture for such a soft-output operation. The principle of the method is given by Battail [2] or by Hagenauer-Hoeher [3], who effectively suggest very similar algorithms, but which do not directly lend themselves to straightforward hardware implementation. The modifications we propose do not alter the quality of the weighting and lead to a real-time circuit, the size of which is only twice the size of the conventional Viterbi decoder.

2. The basic weighting algorithm

In this presentation, we suppose that the reader is familiar with the Viterbi algorithm that we consider only in the case of binary codes, for which the decoder has to select one of two paths, at each of the 2^v nodes, where v is the code memory. The encoder input at time k is denoted d(k+v) and the output of the encoder memory (shift register) is d(k).

The fundamental information for allocating a reliability value to the choice of one path (the so-called survivor), of the two paths which merge in a node, is the difference between the two accumulated metrics associated with this node. Let us call $M_s(k,m)$, the accumulated metric of the survivor at time k and node m, and $M_c(k,m)$, the accumulated metric of the other path coming at the same time to node m, which is called the concurrent path. The greater the difference $M_c(k,m)-M_s(k,m) \geq 0$, the more reliable the selection of the survivor at time k and node m. Because the two paths give two opposite binary decisions about d(k), $M_c(k,m)-M_s(k,m)$ can be directly assigned as a first estimation of the weight of the decision given by the survivor about d(k), conditionally to the choice of node m. If s(k,m) is the sign ($\pm$ 1) of this decision, its weighted value a(k,m) is

$$a(k,m) = s(k,m).(M_c(k,m)-M_s(k,m)). \qquad (1)$$

On condition that weights |a(k,m)| are memorized for each state m, in addition to the path memory which provides s(k,m), the weighted decision is available at the output of the decoder, after the trace-back operation and a latency L, where L is the truncation length of the trellis. However, this weighting method is greatly perfectible as shown below.

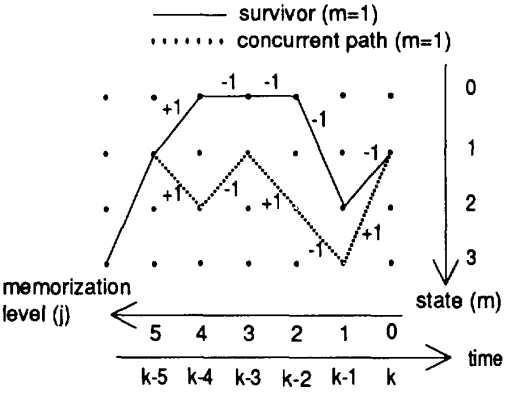

Fig. 1. Example of survivor and concurrent path in a 4-state trellis.

Let us consider the example in Fig. 1, of a 4-state trellis. k is the current time at the input of the trellis and j is the memorization level (or depth) corresponding to decision time k-j. In this example, the survivor and concurrent paths considered at state m=1 and time k, have diverged at time k-5. The two paths give opposite binary decisions at times k and k-2. Let us suppose, for instance, that the weight memorized on the survivor is high at level j=2 (time k-2), and that $M_s(k,1) = M_c(k,1)$. Then, the weight at time k will justifiably be 0 but the weight at time k-2 is no longer realistic, because the concurrent path could also have been the survivor, with an opposite binary decision! Therefore, an appreciable

0-7803-0950-2/93/$3.00©1993IEEE

improvement of the weighting method consists in performing 2^V parallel trace-back operations, starting from each node m at j=0, and revising or updating the values which have been memorized on the 2^V survivors, by taking into account the present values |a(k,m)|.

Let us denote $a_j(k,m)$ and $a_j'(k,m)$ the weighted decisions which have been memorized at level j, on the survivor and on the concurrent path respectively, both returning from node m at j=0 and time k. These decisions are then relative to time k-j.

$$a_j(k,m) = s_j(k,m).|a_j(k,m)|$$
$$a_j'(k,m) = s_j'(k,m).|a_j'(k,m)|, \quad j = 1, ..., L. \quad (2)$$

According to the probabilistic calculation performed by Battail, the revised value $\mathbf{a_j(k,m)}$ of $a_j(k,m)$ is given by

$$\begin{aligned}\mathbf{a_j(k,m)} = \text{Log}\,[&\exp(a_j'(k,m))+\exp(a_j(k,m)+a_j'(k,m))\\ &+\exp(a_j(k,m)+|a(k,m)|)\\ &+\exp(a_j(k,m)+a_j'(k,m)+|a(k,m)|)]\\ -\text{Log}\,[&1+\exp(a_j(k,m))+\exp(|a(k,m)|)\\ &+\exp(a_j'(k,m)+|a(k,m)|)]. \quad (3)\end{aligned}$$

This expression can be simplified without great alteration, in the form

$$\begin{aligned}\mathbf{a_j(k,m)} = \max\,[&a_j'(k,m), a_j(k,m)+a_j'(k,m),\\ &a_j(k,m)+|a(k,m)|,\\ &a_j(k,m)+a_j'(k,m)+|a(k,m)|]\\ -\max\,[&0, a_j(k,m), |a(k,m)|,\\ &a_j'(k,m)+|a(k,m)|]. \quad (4)\end{aligned}$$

It is again possible to continue simplification by splitting up the latter expression into two cases, depending on the sign of $s_j(k,m).s_j'(k,m)$:

case 1 : $s_j(k,m).s_j'(k,m) < 0$
$\mathbf{a_j(k,m)} = s_j(k,m)\,.\,\min\,[|a_j(k,m)|, |a(k,m)|]$ (5a)

case 2 : $s_j(k,m).s_j'(k,m) > 0$
$\mathbf{a_j(k,m)} = s_j(k,m)\,.\,\min\,[|a_j(k,m)|, |a(k,m)|+|a_j'(k,m)|]$. (5b)

Fortunately, the revision does not seem to feel the effects of neglecting case 2. Indeed, when the two paths give the same binary decision at level j>0, revision is not of great importance and we can only process the first case, for which the knowledge of $|a_j'(k,m)|$ is no longer necessary. This additional and in practice important simplification of expression (3) leads to a revision procedure close to the one proposed by Hagenauer-Hoeher. If the output of the decoder is quantized on n bits (one for the sign and n-1 for the weight), the revision operation, besides the classical circuits of the Viterbi decoder, requires:

- an $L.2^V.(n-1)$ bit memory for the storage of $|a_j(k,m)|$, j=1,...,L, m=1,...,2^V.
- $L.2^V$ binary comparators of $s_j(k,m)$ and $s_j'(k,m)$,
- $L.2^V$ numerical comparators and multiplexers for the possible minimization of $|a_j(k,m)|$ by |a(k,m)|, on n-1 bits.

All this really makes for a cumbersome decoder, with an impractical format for the memory which, moreover, must be multi-accessed in one data-clock period. These drawbacks can be eliminated by adopting the *a posteriori* weighting algorithm described in the next chapter.

3. The *a posteriori* weighting algorithm

The complexity of the algorithm, as set out above, comes mainly from the necessity to revise memorized data over the 2^V survivor paths, because the final survivor (the maximum likelihood path) is not yet known. This difficulty is removed if the weighting operation is performed after the final survivor is completely (or almost completely) established, that is to say if the weighting procedure is preceded by regular Viterbi decoding. Let us reference $T(2^V, L)$ the trellis in which this first decoding step is done, with 2^V states as the vertical dimension and the truncation length L as the horizontal dimension. Let us call m_0 and m_L the nodes of the maximum likelihood path in trellis T at levels j=0 and j=L, at time k. Assuming that L is sufficiently large, node m_L no longer (or not so much) depends on the present and future input symbols of the decoder. Then it is possible to achieve revision in a second trellis, referenced $T'(2^V, L')$, on a single path returning from node m_L, assumed to be the definitive and permanent maximum likelihood path.

Since the maximum likelihood path is assumed not to change in the second trellis T' ($L+1 \le j \le L+L'$), the index m_0 in writing the weights |a| or the signs s, is no longer necessary for $j \ge L$. So, let us denote

$$w_j(k) = |a_j(k,m_0)| \qquad j \ge L$$
$$s_j(k) = s_j(k,m_0) \qquad j \ge L. \quad (6)$$

In order to achieve revision in trellis T', beginning at memorization level j=L+1, the weight $w_L(k)$ has to be known. There are two means to make this information available : memorization of all |a(k,m)| computed at level j=0, like in the basic algorithm, or the repeated calculation of metrics at level j=L, L clock periods later, providing that input symbols have been brought as far as level j=L, by a delay line. The latter solution, which discards the need for an unusual format and multi-access memory, is quite preferable for simplicity and high data rate and requires only one subtraction between the two accumulated metrics which have been computed again at level j=L and state m_L. Then $w_L(k)$ can be stored as the temporary weight associated with the binary decision $s_L(k)$ relative to time k-L, in a shift register. At the same time, $w_L(k)$ acts as the updating weight for all the contents of this same shift register, that is to say for the L' previous weights associated with the binary decisions of the maximum likelihood path considered in trellis T'. These weights are revised according to the procedure given by (5a), over the single survivor returning from state m_L at level j=L.

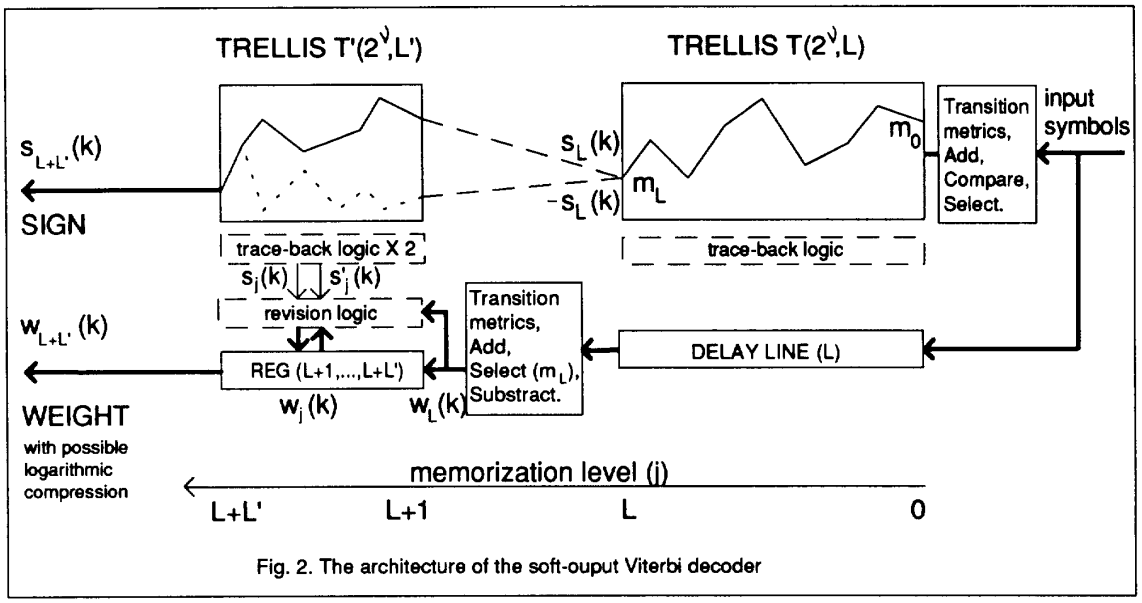

Fig. 2. The architecture of the soft-ouput Viterbi decoder

4. The architecture of the soft-output Viterbi decoder

The organization of the decoder is given in Fig. 2. It contains :

- in the first part, a classical Viterbi decoder with trellis T, computation of transition metrics, Add, Compare and Select (ACS) blocks, and the trace-back logic. In addition to these classical circuits, a delay line of length L makes the input symbols also available at the end of T;
- in the second part, the trellis T' in which a double trace-back is performed, the first one as the normal continuation of the trace-back in T, the second one starting from the same state m_L, but over the concurrent path, that is to say the path giving $-s_L(k)$ as the binary decision at level j=L (decision time k-L). From the input symbols which have been carried by the delay line, the transition metrics and the accumulated metrics are calculated a second time; at this level, there is no need to perform the operations of comparison and selection because the results of these are already known and available at the output of T. In order to get the weight $w_L(k)$, it has just to select node m_L, subtract its two accumulated metrics and take the absolute value of the result. $w_L(k)$ is then applied to the input of the shift register called REG, of length L', and concomitantly acts as the revision weight, in the following way. At level j ($L+1 \le j \le L+L'$) $s_j(k)$ and $s_j'(k)$ are the binary decisions yielded by the maximum likelihood path and the concurrent path in T', $w_j(k)$ is the content of REG at level j and time k. Then if $s_j(k)$ and $s_j'(k)$ are different and if $w_j(k) > w_L(k)$, $w_j(k)$ has to be replaced by $w_L(k)$. This operation is done in parallel over the L' values of REG and requires L' binary comparators, numerical comparators and multiplexers. One set of these operators for one level of revision, is described in Fig. 3. The final soft decision of the decoder, relative to time k-L-L', is then given by $s_{L+L'}(k)$ as the sign, and by $w_{L+L'}(k)=REG(L+L')$ as the absolute value.

Depending on the use that is made of the output of the decoder, it may be profitable to apply a logarithmic compression to the final weight so as to offset the possible defects of the non-optimal weighting operation, resulting from the drastic simplification of formula (3). This logarithmic compression is noticeably worthwhile when the data is used as the input of another Viterbi decoder.

Since the weighting procedure is done in a serial and pipe-line manner with the usual Viterbi decoding, it does not at all lower the speed of the classical decoder. In other respects, it does not seem necessary to give to each of the truncation lengths L and L' the same value as in the binary decoder (i.e. 4 or 5 times the constraint length K=v+1 for a 1/2 coding rate). Very good results can be obtained, for instance, when L' is half the value of L, so that the soft-output decoder, taken as a whole, can restrict its size to less than twice that of the classical Viterbi decoder.

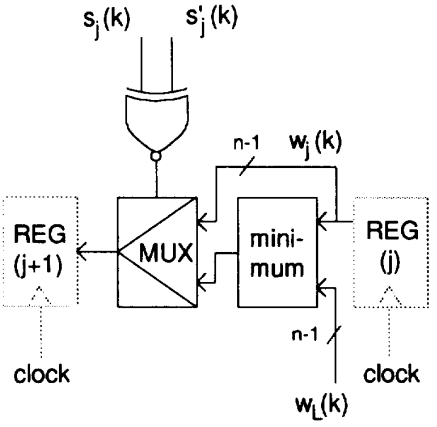

Fig. 3. Revision logic at level j

5. An application to the decoding of the concatenation of two convolutional codes

So as to estimate the quality of the *a posteriori* weighting method, we consider an application of concatenating two convolutional codes and decoding it after transmission over a memoryless additive white gaussian noise (AWGN) channel with binary modulation. The information bits are first coded by a punctured encoder of rate 7/8 with ν=2. The symbols stemming from this first encoder are interleaved and re-encoded by a second punctured encoder of rate 4/7 with ν=4. The global rate of the composite code is then 1/2. The complete decoder is made up of a first or inner 16-state soft-output decoder and a second or outer 4-state classical Viterbi decoder, separated by a de-interleaver.

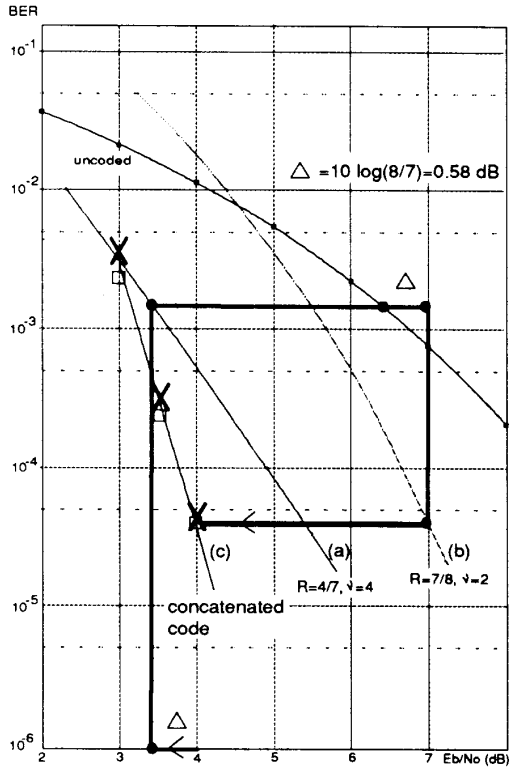

Fig. 4.
Binary error rate given by the decoding of concatenation of (R=7/8,ν=2) and (R=4/7,ν =4) codes over a gaussian channel.
(c): under the AWGN soft-output inner decoder assumption,
X: with the soft-output Viterbi decoder as the inner decoder,
□ : with the Bahl et al. algorithm as the inner decoder.

As a reference, we first graphically determine the binary error rate (BER) at the ouput of the global decoder, assuming that the inner decoder is able to provide ideally AWGN soft-output information, after de-interleaving. This is done in Fig. 4. Curves (a) and (b) here give the BER at the output of each of the two elementary decoders, of rates 4/7 and 7/8, considered individually and using the classical Viterbi algorithm, as a function of the signal to noise ratio (SNR) E_b/N_0. In order to obtain the BER of the concatenated decoder under the assumption of an AWGN soft-output inner decoder, we proceed as follows. Let us choose, for instance, E_b/N_0=4 dB. Because the rate of the inner code is 4/7 and the global rate is 1/2, the abscissa must be lowered by Δ=10 log((4/7)/(1/2))=10 log(8/7)=0.58 dB. Curve (a) then supplies the BER at the ouput of the inner decoder, which corresponds to a certain SNR (about 6.4 dB) on the uncoded transmission curve. Because this SNR involves the energy per symbol and not the energy per information bit, Δ must be added to this value and finally the sought-for BER is given by curve (b) at E_b/N_0 ≈7 dB. Curve (c) embodies this linked graphic operation for any SNR between 3 and 4.25 dB.

Now, we perform a simulation of the global decoder by using the *a posteriori* weighting algorithm for the inner decoder, with L=24 and L'=15, and the regular Viterbi algorithm for the outer decoder, with L=56. A logarithmic compression, of the form

$$y = \theta \, \mathrm{Log}(1+x/\theta), \qquad (7)$$

where θ is twice the mean absolute value of the input samples, is used in the weighting procedure. The results of this simulation are identified by crosses (X) in Fig. 4 and are very close to curve (c), demonstrating the good quality of the *a posteriori* weighting method. Another type of simulation has been done for comparison, with the Bahl *et al.* algorithm [4] as the inner decoder principle. This probabilistic algorithm provides the optimal estimate of the likelihood of the data coded by a convolutional code, but unfortunately requiring high complexity processing and not real-time. The results are indicated by squares and are slightly better for low SNR, even in comparison with the reference curve (c), which seems to show that the distribution relationship of output samples is not gaussian when calculated by the optimal algorithm. Nevertheless, it does not give much better results than the method described in this paper, which has the decisive advantage of real-time operation and relative simplicity.

Acknowledgement

The authors wish to thank Pr. G. Battail for his helpful explanations about the basic weighting algorithm and P. Thitimasjhima who programmed the Bahl *et al.* decoding.

References

[1] G. D. Forney, "The Viterbi algorithm," Proc. IEEE, vol. 61, N° 3, pp. 268-278, Mar. 1973.

[2] G. Battail, "Pondération des symboles décodés par l'algorithme de Viterbi," (in French), Ann. Télécommun., Fr., 42, N° 1-2, pp. 31-38, Jan. 1987.

[3] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision ouputs and its applications," Proc. of IEEE Globecom '89, Dallas, Texas, pp. 47.11-47.17, Nov. 1989.

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," IEEE Trans. Inform. Theory, vol. IT-20, pp. 284-287, Mar. 1974.