A New Architecture For The Fast Viterbi Algorithm

Inkyu Lee† Jeff L. Sonntag‡

†Lucent Technologies, Bell Laboratories
Murray Hill, NJ 07974
‡Lucent Technologies, Bell Laboratories
Allentown, PA 18103

Abstract

A novel architecture design to speed up the Viterbi algorithm is proposed. By doubling the number of states in the trellis, the serial operation of a traditional Add-Compare-Select (ACS) unit is transformed into a parallel operation, thus achieving a substantial speed increase. The use of the proposed architecture would increase the speed by 33% at the expense of a faily modest increase in area, thus removing the Viterbi detector/decoder from the worst case speed bottleneck path in most high-speed applications. A simple example is shown to illustrate the proposed algorithm in Maximum Likelihood Sequence Detector.

1 Introduction

The Viterbi algorithm was first introduced in 1967 as a means to decode convolutional codes [1]. And later Forney [2] showed that the Viterbi algorithm can be applied to implementing a Maximum Likelihood Sequence Estimation (MLSE). Since then, the Viterbi algorithm has been widely used in many communication systems in both ML convolutional decoder and ML sequence detector.

Many high speed applications adopt the Viterbi algorithm operating at several hundreds Mbps and this operating clock speed has been increasing constantly. These examples include the ML sequence detector in magnetic recording systems and convolutional decoders for error correction. So there has been a strong need to achieve a higher speed and this motivates the work presented here.

The main operation unit performing the Viterbi algorithm is called an Add-Compare-Select (ACS) unit. However, due to the feedback loop, the ACS unit is considered as the bottleneck in actual implementation of high speed applications.

Several architectures have been proposed to speed up the Viterbi algorithm operation. In [3, 4], they increased the speed throughput of the Viterbi algorithm by having multiple ACS units in a parallel implementation, thus trading area for speed. In their approaches, the main structure inside the ACS unit remains the same. Black *et al* [5] also extended the parallel processing by applying one stage of lookahead to both the ACS and trace-back recursions, resulting in a radix-4 ACS and a radix-16 trace-back iteration.

In this paper, we propose a new architecture of the ACS unit [6] that speeds up the Viterbi algorithm by doubling the states of the trellis. By reformulating the Viterbi algorithm, the proposed architecture provides an alternative approach to high throughput design. We will refer to the proposed architecture as the "double state". By having the "double state", a serial operation of the ACS unit can be transformed to a parallel operation. This new approach enables us to break the ACS speed bottleneck with a fairly modest increase of area.

This paper is organized as follows. Section 2 briefly describes the Viterbi algorithm and the ACS unit. The new architecture for the fast Viterbi algorithm is proposed in Section 3. In Section 4, a simple example for the proposed architecture is provided for Maximum Likelihood Sequence Detector, and projection for area and speed is presented. Also a brief comparision between the proposed architecture and the parallel processing approach introduced in [5] is made. Finally, Section 5 discusses the summary.

2 The Viterbi Algorithm

In this section, we will briefly explain the Viterbi algorithm and also introduce the notations used throughout this paper.

Consider the Maximum Likelihood Sequence Detector case first. Assuming the channel response polynomial $H(D)$ is given where D denotes a delay operator, the Viterbi algorithm recursively optimizes the most-likely

0-7803-6451-1/00/$10.00 © 2000 IEEE

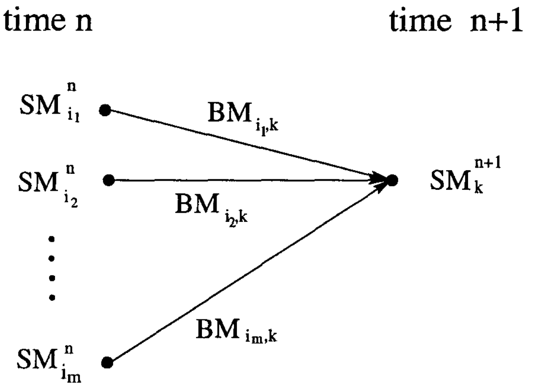

Figure 1: The state metric update

path by accumulating the branch metric (BM) for each state where the number of states is determined by m^N. Here m represents the size of the input alphabet and N denotes the channel memory length. BM for each transition in a trellis is computed using $H(D)$ and its trellis input corresponding to the transition.

In each state of a trellis, the previous state metric and the corresponding branch metric are added together and then the accumulated State Metric (SM) is updated by choosing the minimum of all possible cases recursively:

$$SM_k^{n+1} = \min_i (SM_i^n + BM_{i,k}^n)$$

where SM_i^n represents the state metric of the i th state at time n and $BM_{i,k}^n$ denotes the branch metric at time n associated with a transition from the i th state to the k th state. Figure 1 illustrates this update. The minimization at the k th state is carried out for all possible previous states $i_1, i_2, \cdots i_m$.

This update operation is performed in the Add-Compare-Select (ACS) unit in the Viterbi Algorithm. Each operation in the ACS unit is carried out in a purely serial way, thus causing the worst speed bottleneck in the whole throughput. This is clearly illustrated in Figure 2 for a binary input case. In this diagram, a triangle and a trapezoid represent a comparator and a multiplexer respectively. The multiplexer chooses either an input at the top or at the bottom as an output depending on the middle input at the left side. In the following section, we propose a new architecture to speed up the ACS unit by changing the serial ACS unit into a parallel structure.

3 The New Structure: Double-State

For simplicity, we assume a binary input case ($m = 2$). This result can be easily generalized to a multiple input

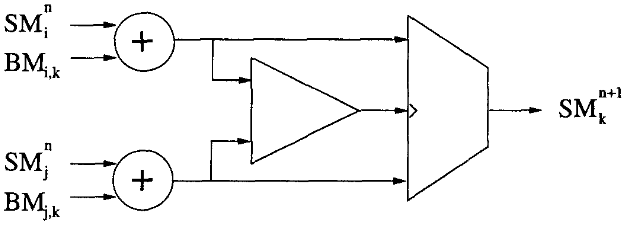

Figure 2: The Add-Compare-Select unit for the Viterbi algorithm

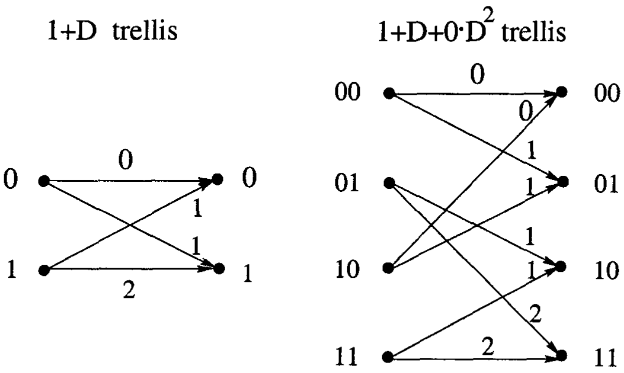

Figure 3: Two equivalent trellises

level case. Also, we continue explaining a new structure in the ML sequence detector case. Applying the same structure to the ML convolutional decoder is straightforward.

First, note that the channel response polynomial $H(D)$ of order N could be written as: $H(D) = h_0 + h_1 D + \cdots + h_M D^N + 0 \cdot D^{N+1}$. As an example, Figure 3 illustrates two equivalent trellises for the 2 state $1 + D$ channel and the 4 state $1 + D + 0 \cdot D^2$ channel. The numbers next to the states represent the input sequence. (For example, 10 of the left hand side state of the $1 + D + 0 \cdot D^2$ trellis represents an input 1 and 0 at time $n - 2$ and $n - 1$ respectively.) Also, the numbers on the arrow line show the ideal channel output associated with the transition.

For a channel $H(D)$ which has a zero coefficient for the last coefficient, the branch metrics for two transitions which have the same ending state are same, because the two starting states are different in only the oldest bit position. In this case, the Viterbi processor has 2^{N+1} states, even though $H(D)$ is actually a polynomial of order N (thus the term "double state"). By having the "double state" in a trellis, the brach metrics ending in one state are all the same. This means that when choosing the

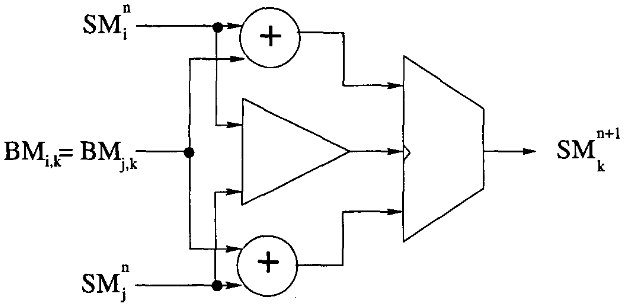

Figure 4: A new Add-Compare-Select unit

minimum of two possible state metrics $SM_i^n + BM_{i,k}$ and $SM_j^n + BM_{j,k}$, we can select the less of two previous state metrics SM_i^n and SM_j^n without waiting for an addition of the branch metrics. (In this case, $BM_{i,k} = BM_{j,k}$) Equivalently, we perform the following recursion:

$$SM_k^{n+1} = \min_i(SM_i^n) + BM_k^n$$

For example, in the current state 00 of Figure 3, two incoming paths from the previous states 00 and 10 have the same branch metric 0. This applies to all the other states, since in the double state the oldest input to the Viterbi processor makes no contribution on computing the branch metric for each state transitions. Therefore, in the double state structure, the "Add" operation which computes the state metric can be carried out at the same time as the "Compare" operation. This new structure is shown in Figure 4. As clearly shown in this diagram, two branch metrics $BM_{i,k}$ and $BM_{j,k}$ are the same.

A careful investigation of the double state trellis reveals that a further hardware savings is possible. Looking at the current states 00 and 01 in Figure 3, they share the same pair of the previous states 00 and 10. Therefore, if the current state 00 chooses the path from the previous state 10 over one from the previous state 00, then the same decision is made at the "Select" operation for the current state 01. This is the same for the other pair of the states 10 and 11. Thus, every two states in the double state structure can share the same decision making unit in their "Compare" operation. This can be easily generalized to an m-ary input case. Combining two states which share the same previous states in the double state, the new ACS structure is illustrated in Figure 5. In this diagram, the state $k1$ and $k2$ share the comparator, thus reduce the hardware complexity. As a result, 2^N units of the ADC shown in Figure 5 are used in the double state architecture.

At first it appears that the double state trellis in Figure 3 requires twice as much computation for the state metric than the ordinary trellis. However, it shall be

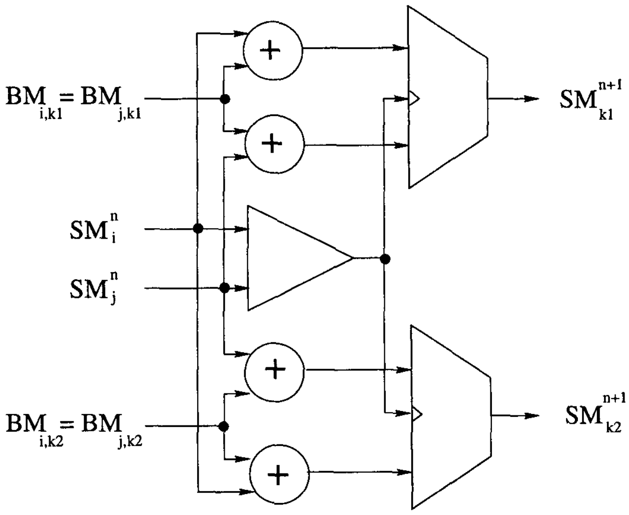

Figure 5: A combined Add-Compare-Select unit

shown in the following section that there is no redundant information in the double state trellis. Actually, every transition shown in the ordinary trellis should be computed, whereas only the half of the transitions shown in the double state trellis are used for computation.

4 Example and Area-Speed Analysis

This section shows a simple example explaining that the double state structure contains no redundancy at all. Also, area and speed estimate analysis is given.

4.1 Example

Figure 6 compares the ordinary Viterbi algorithm and the double state in a binary $1 + D$ channel shown in Figure 3. Here y represents the input sequence to the Viterbi processor. The branch metric is computed using the normalized equation $-y \cdot \hat{y}/2 + \hat{y}^2/4$ where $\hat{y}$ shows the ideal channel output. The thick line and the dashed lines indicate the survival path and the discarded paths respectively.

Here in this example, it is readily apparent that the two representations are exactly the same. Each transition in the ordinary trellis appears in the double state trellis. In the ordinary trellis the discarded paths are not shown conventionally, while in the double state trellis representation there is no hidden discarded paths. The only difference in two trellises is that the decision made in the double state has one more latency. For example, at time $n = 4$, the state 0 in the ordinary trellis chooses a path

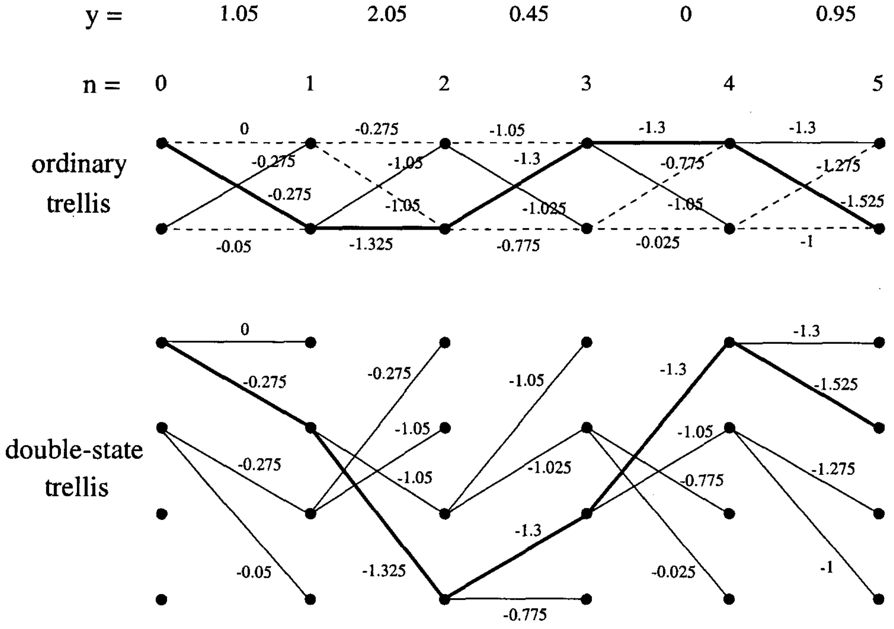

Figure 6: Example of the double state

with metric -1.3 over one with metric -0.775. In contrast, the double state trellis makes the same decision at time $n = 5$. This extra latency can be, however, corrected by noting that knowledge of a current sample value is not necessary to make a decision. A simple correction rule is described in [6]. Another point to note is that the double state trellis does not make any decision at the initial stage (time $n = 0$) so that transitions shown at time $n = 0$ can be arbitrarily made and this first decision is neglected.

4.2 Area-Speed Analysis

Compared to the ordinary Viterbi algorithm, the proposed double state structure requires twice more adders, state metric registers and multiplexers. Everything else remains the same including the path memory, since the number of the surviving paths in the double state trellis is the same as that in the ordinary trellis. As the adders, state metric registers and multiplexers are a substantial, but not a dominant portion of the area of the ordinary Viterbi processor, the expected area in the double state implementation is roughly 50% more than the ordinary implementation. In some applications, the Viterbi processor comprises of only a small part of the complete chip. For example, in an 8 state EPR4 Viterbi detector chip [7] which includes a timing recovery, adaptive equalizer, continuous time filter, encoder/decoder and servo processing, the Viterbi detector would take only about 8% of the total area of the chip, thus the area increase for the double state approach is only about 4% of the chip area in this case.

Transistor level simulations of an ACS designed for high speed operation show that of a clock cycle, about 50% is used by the add, 25% by the compare operation, 5% by the multiplexing, and 20% by the register setup and propagate delays. In the proposed double state structure, the propagation to the adder can be operated at the same time as the comparator delay, thus saving 25% of the clock cycle. Therefore, the speed of the double state should be 33% faster than the ordinary ACS unit.

A comparative synthesis for the 16 state ML sequence detector with a fixed channel response polynomial $H(D)$ was made [8] to compare the speed and area estimate for the proposed architecture and the parallel process approach. This demonstrated that the "double state" architecture provides 30% throughput increase with 32% additional area over the conventional implementation, whereas the synthesis based on the parallel processing yields 58% speed-up with 151% area increase. It was also noted that for applications where $H(D)$ is programmable [9], the area increase for the parallel processing technique is much higher due to the increased complexity in 4-way adders and comparators.

It should be noted that the proposed double state architecture can replace the ordinary ACS unit in other fast architectures [3, 4, 5] and further speed up the Viterbi processor operation.

5 Conclusions

We have proposed a new ACS unit which breaks the speed bottleneck in the Viterbi algorithm. By doubling the states of the ordinary trellis, the serial operation in the ACS unit is reorganized so that the "Add" and "Compare" operations are carried out at the same time, thus increasing the speed of the operation cycle. It is projected that 33% of speed up can be achieved. Especially, following a "system on a chip" trend, the increased area becomes negligible compared to the whole chip size.

Compared with the parallel processing technique, the proposed architecture provides a good trade-off between speed and area, and is better suited for an area intensive application.

The use of the proposed double state architecture will remove the Viterbi processor from the speed bottleneck path in many communication application VLSI chip designs at a fairly modest cost in area and power dissipation.

Acknowledgment

The authors would like to thank J. Garofalo for providing simulation results of several Viterbi processor architectures.

References

[1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[2] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Transactions on Information Theory*, vol. IT-18, pp. 363–378, May 1972.

[3] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Transactions on Communications*, vol. COM-37, pp. 785–790, August 1989.

[4] H. K. Thapar and J. M. Cioffi, "A block processing method for designing high-speed Viterbi detectors," in *Proc. of International Conference on Communications*, pp. 1096–1100, 1989.

[5] P. J. Black and T. H. Meng, "A 140Mb/s, 32 state, radix 4 Viterbi decoder," *IEEE Journal of Solid-state Circuits*, vol. 27, pp. 1877–1885, December 1992.

[6] I. Lee and J. L. Sonntag, "A new architecture for the fast Viterbi algorithm," patent filed, March 1998.

[7] J. Fields and et al, "Design of a high-speed low power EPR4 read channel chip," in *Proc. of ISSCC*, 1996.

[8] J. Garofalo, private communication, Lucent Technologies, 2000

[9] H. L. Lou, *The study and design of a programmable processor for Viterbi detection.* PhD thesis, Stanford University, Stanford, CA 94305, December 1992.