A New Architecture for the Fast Viterbi Algorithm

Inkyu Lee, *Senior Member, IEEE*, and Jeff L. Sonntag

***Abstract*—A novel architecture design to speed up the Viterbi algorithm is proposed. By increasing the number of states in the trellis, the serial operation of a traditional add–compare–select unit is transformed into a parallel operation, thus achieving a substantial speed increase. The proposed architecture would increase the speed by 33% at the expense of a faily modest increase in area, thus becoming an attractive approach in high-speed applications. A simple example is shown to illustrate the proposed algorithm in maximum-likelihood sequence detector. A comparative synthesis is made to compare the proposed architecture with other approaches, and synthesis simulations confirm the projection of the throughput gain. Also, the proposed algorithm is extended to the block-processing architecture, and we show that an additional 50% speedup is achieved.**

***Index Terms*—Detector/decoder, fast architecture, Viterbi algorithm (VA).**

I. Introduction

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

Many high-speed applications adopt the VA operating at several hundreds megabits per second, and this operating clock speed has been increasing constantly. These examples include the ML sequence detector (MLSD) 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 VA 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 for speeding up the VA operation. In [3], they increased the speed throughput of the VA by having multiple ACS units in a parallel implementation, thus trading area for speed. With similar approaches, Thapar *et al.* [4] and Black *et al.* [5] also extended the block processing by applying one stage of lookahead to both the ACS and traceback recursions, resulting in a radix-4 ACS and a radix-16 traceback iteration. This architecture is referred to as "block processing."

In this letter, we propose a new architecture of the ACS unit [6] that speeds up the VA by increasing the states of the trellis. By reformulating the VA, 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 speed up the ACS operation with a fairly modest increase of area. We also extend the double-state technique to the block-processing architecture. The overall speed analysis shows that when the double-state technique is applied to the conventional Viterbi processor and the block processor, a speedup factor of 33% and 50% is expected, respectively.

This letter is organized as follows. Section II briefly describes the VA and the ACS unit. The new architecture for the fast VA is proposed in Section III. In Section IV, a simple example for the proposed architecture is provided for MLSE, and projection for area and speed is presented along with the actual synthesis data which confirms the estimates. Also, a comparison between the proposed architecture and the block-processing approach is made. In Section V, we extend the proposed double-state approach to the block-processing technique. Finally, Section VI discusses the summary.

II. Viterbi Algorithm (VA)

In this section, we will briefly explain the VA and also introduce the notations used throughout this letter.

Consider the MLSD case first. Assuming the channel response polynomial $H(D)$ is given where D denotes a delay operator, the VA recursively optimizes the most likely 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 (SM) and the corresponding BM are added together, and then the accumulated SM is updated by choosing the minimum of all possible cases recursively

$$\mathrm{SM}_k^{n+1} = \min_i(\mathrm{SM}_i^n + \mathrm{BM}_{i,k}^n)$$

where SM_i^n represents the SM of the ith state at time n, and $\mathrm{BM}_{i,k}^n$ denotes the BM at time n associated with a transition from the ith state to the kth state. Fig. 1 illustrates this update. The minimization at the kth state is carried out for all possible previous states $i_1, i_2, \cdots i_m$.

Paper approved by R. D. Wesel, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received October 6, 2000; revised June 18, 2002. This work was supported in part by a Korea University grant, and in part by the Korea Science and Engineering Foundation under Grant R08-2003-000-10761-0.

I. Lee is with the Department of Communications Engineering, Korea University, Seoul, Korea (e-mail: inkyu@korea.ac.kr).

J. L. Sonntag is with Accelerant Networks, Beaverton, OR 97006 USA (e-mail: jsonntag@accelerant.net).

Digital Object Identifier 10.1109/TCOMM.2003.818100

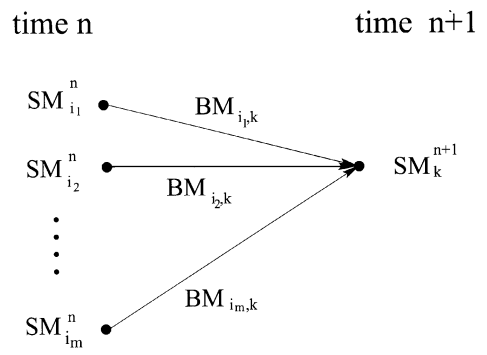

Fig. 1. SM update.

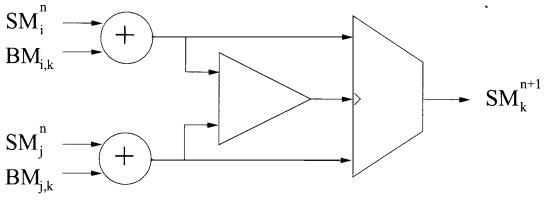

Fig. 2. ACS unit for the VA.

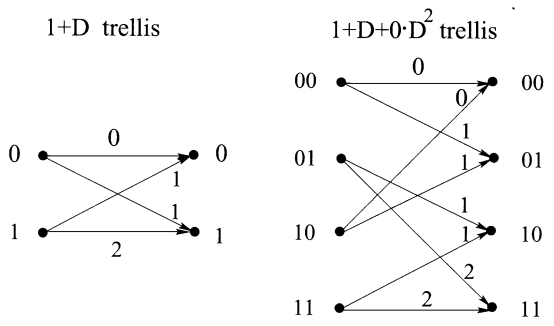

Fig. 3. Two equivalent trellises.

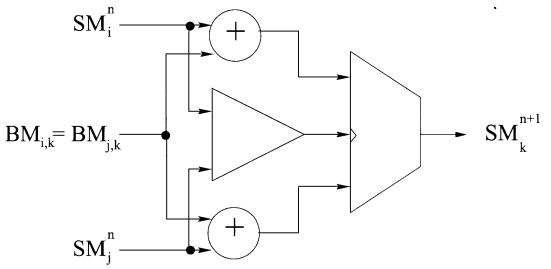

Fig. 4. A new ACS unit.

This update operation is performed in the ACS unit in the VA. 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 Fig. 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.

III. New Structure: Double State

For simplicity, we assume a binary input case ($m = 2$). This result can be easily generalized to a multiple-input level case. Also, we continue explaining a new structure in the MLSD 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 $H(D) = h_0 + h_1 D + \cdots + h_N D^N + 0 \cdot D^{N+1}$. As an example, Fig. 3 illustrates two equivalent trellises for the two-state $1 + D$ channel and the four-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 BMs for two transitions which have the same ending state are the 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 BMs ending in one state are all the same. This means that when choosing the minimum of two possible SMs $\mathrm{SM}_i^n + \mathrm{BM}_{i,k}$ and $\mathrm{SM}_j^n + \mathrm{BM}_{j,k}$, we can select the less of two previous SMs SM_i^n and SM_j^n without waiting for an addition of the BMs. (In this case, $\mathrm{BM}_{i,k} = \mathrm{BM}_{j,k}$) Equivalently, we perform the following recursion:

$$\mathrm{SM}_k^{n+1} = \min_i(\mathrm{SM}_i^n) + \mathrm{BM}_k^n.$$

For example, in the current state 00 of Fig. 3, two incoming paths from the previous states 00 and 10 have the same BM 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 BM for each state transitions. Therefore, in the double-state structure, the "Add" operation which computes the SM can be carried out at the same time as the "Compare" operation. This new structure is shown in Fig. 4. As clearly shown in this diagram, two BMs $\mathrm{BM}_{i,k}$ and $\mathrm{BM}_{j,k}$ 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 Fig. 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 Fig. 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 Fig. 3 are used in the double-state architecture.

At first it appears that the double-state trellis in Fig. 3 requires twice as much computation for the SM than the ordinary trellis. However, it shall be shown in the following section that the

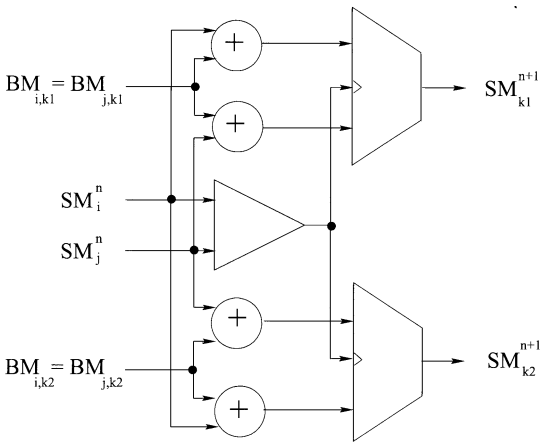

Fig. 5. A combined ACS unit.

double-state architecture computes no redundant information. 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.

Note that the proposed architecture can be explained in terms of a simple lookahead with nonlinearities in the loop [7], since it is viewed as retiming of the existing trellis update. Basically, the proposed architecture transforms the conventional ACS unit into a "compare–add–select" operation. In contrast, the differential trellis decoding (DTD) algorithm has been proposed in [8] based on a "compare–select–add" method. While DTD reduces the number of additions by increasing the number of preprocessing operations in the trellis, the double-state architecture speeds up the operation by increasing the number of states in the trellis. Therefore, the double-state approach is better suited for the high-speed applications where additional complexity is acceptable.

IV. 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 as well as the actual synthesis data.

A. Example

Fig. 6 compares the ordinary VA and the double state in a binary $1 + D$ channel shown in Fig. 3. Here, y represents the input sequence to the Viterbi processor. The BM 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 are no hidden discarded paths. The only difference in the 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 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, however, be corrected by noting that knowledge of a current sample value is not necessary to make a decision. 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.

B. Area-Speed Analysis

Compared to the ordinary VA, the proposed double-state structure requires twice more adders, SM 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, SM 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. However, this area estimate assumes the worst-case scenario, and it can be lower than that in actual designs. This is confirmed at the end of this section. In some applications, the Viterbi processor comprises only a small part of the complete chip. For example, in an eight-state EPR4 Viterbi detector chip [9] 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 MLSD with a fixed-channel response polynomial $H(D)$ was made [10] to compare the speed and area estimate for the proposed architecture and the block process approach, and this result is summarized in Table I. 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 block processing yields 58% speed-up with 151% area increase. It is clear that while the throughput gain of the proposed architecture is smaller than the block approach, the former requires a much smaller area than the latter. It was also noted that for applications where $H(D)$ is programmable [11], the area increase for the block-processing technique is much higher due to the increased complexity in the BM computations. It should also be mentioned that the area and speed comparison presented in Table I may vary, depending on the implementation. Thus, it provides a relative figure of merit in terms of the area and speed.

V. Extension to the Block-Processing Technique

In this section, we extend the proposed double-state architecture to the block-processing technique described in [3]–[5] and show that the Viterbi processor throughput can increase even further.

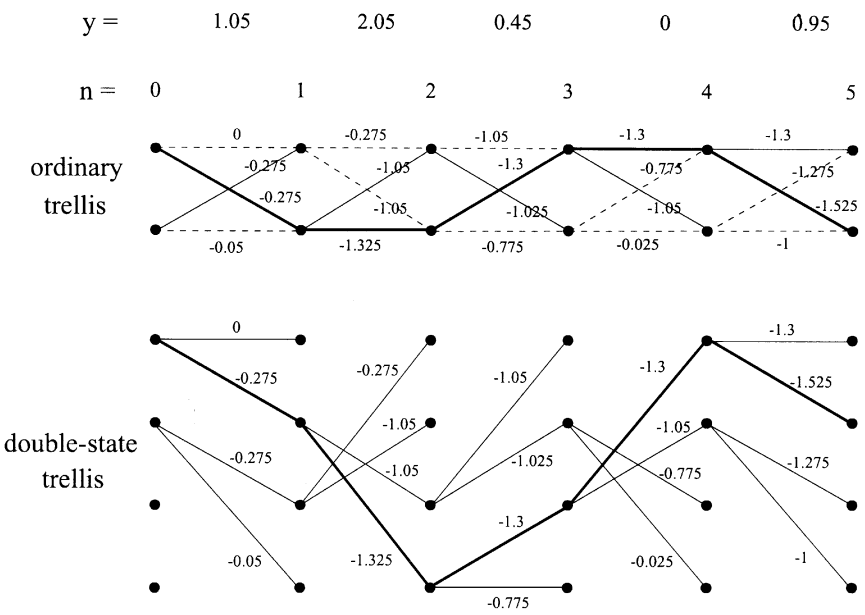

Fig. 6. Example of the double state.

TABLE I
AREA AND SPEED COMPARISON WITH THE BLOCK PROCESSING TECHNIQUE

	double state architecture	parallel process architecture
throughput gain	1.30	1.58
area increase	1.32	2.51

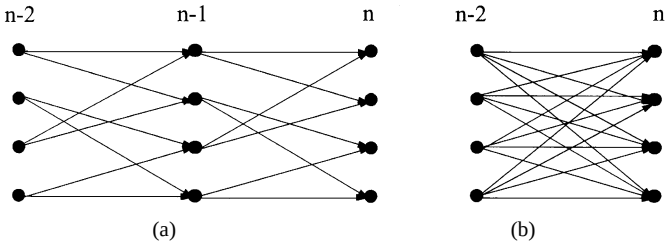

Fig. 7. (a) Four-state trellis. (b) Combined trellis for block processing.

Consider a four-state trellis, described in Fig. 7(a). By processing a multiple-stage trellis at one cycle, the block processing can speed up the whole processing operation. Fig. 7(b) shows a merged two-stage trellis from time $n-2$ to n. In this example, the number of inputs to each ending state is four, and the ACS unit implementing this merged trellis is illustrated in Fig. 8. Note that this ACS unit operates at the half rate, compared to the conventional Viterbi processor, and the BM is computed by adding two separate BMs in each stage trellis.

We can improve the throughput even further by extending the proposed double-state approach to this block-processing architecture. This can be done by adding two zero terms at the end of the channel response polynomial $H(D)$. Then, all four inputs to any ending state have the same BM values, and this allows us to employ the same principle of the double-state technique described in Section III. Following the similar approaches used in the double-state technique, which combines states sharing the same starting states, the resulting architecture is illustrated

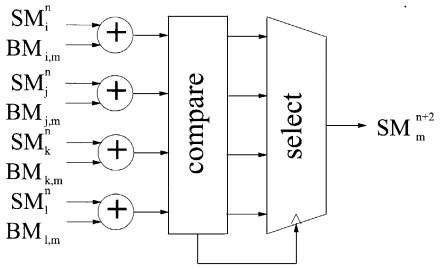

Fig. 8. Block-processing architecture.

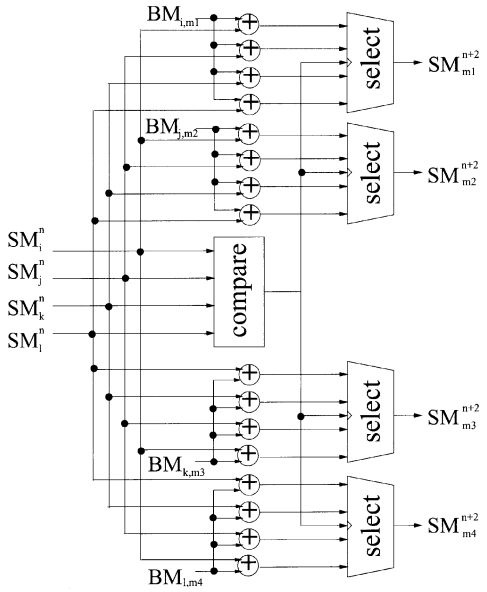

Fig. 9. Double-state architecture combined with the block-processing technique.

in Fig. 9. Note that we need 2^N of these ACS units to implement a 2^N state trellis. The number of adders and multiplexers is quadrupled in this case, and the total complexity is certainly much greater than the block-processing architecture.

As for the throughput gain, it is expected that the double-state approach benefits more when it is applied to the block-processing trellis rather than to the standard Viterbi processor, since the four-input compare operation in the block-processing technique takes longer than the normal two-input compare operation. It is estimated that the adder and the four-input comparator take up about 40% and 35% within one processing cycle, respectively. Therefore, when the double-state technique is combined with the block-processing architecture, the new architecture results in a speedup factor of 1.5 over the block-processing technique. Based on the throughput gain of 1.6 for the block processing reported in the previous section, this indicates that the overall speedup gain of the combined structure over the conventional Viterbi processor would be 140%. Note that this kind of impressive speedup gain can be achieved with a three-stage block-processing technique employing an eight-input comparator, but the three-stage block processor would be more complex.

VI. Conclusions

In this letter, we have proposed a novel ACS unit which speeds up the VA. By increasing 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. We have presented that 33% and 50% of speedup can be achieved, respectively, when the proposed algorithm is applied to the conventional Viterbi processor and to the block processor, and this is confirmed through the synthesis simulation. Especially, following a "system on a chip" trend, the increased area becomes smaller compared with the whole chip size.

Therefore, we conclude that the proposed double-state architecture is an attractive approach to achieve a high speed in many communication application very large-scale integration (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 Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

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

[3] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS bottleneck," *IEEE Trans. Commun.*, vol. 37, pp. 785–790, Aug. 1989.

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

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

[6] I. Lee and J. L. Sonntag, "An Add–Compare–Select Circuit and Method Implementing a Viterbi Algorithm," U.S. Patent 6 148 431, Nov. 14, 2000.

[7] K. K. Parhi, "High-speed architectures for algorithms with quantizer loops," in *Proc. ISCAS*, 1990, pp. 2357–2360.

[8] M. P. C. Fossorier and S. Lin, "Differential trellis decoding of convolutional code," *IEEE Trans. Inform. Theory*, pp. 1046–1053, May 2000.

[9] J. Fields *et al.*, "A 200 Mb/s CMOS EPRML channel with integrated servo demodulator for magnetic hard disks," in *Proc. ISSCC*, 1997, pp. 314–315.

[10] J. Garofalo, Private Communication, Lucent Technologies, 2000.

[11] H. L. Lou, "The study and design of a programmable processor for Viterbi detection," Ph.D. dissertation, Stanford Univ., Stanford, CA, Dec. 1992.