A Trace-Back Method with Source States and its Application to Viterbi Decoders of Low Power and Short Latency

Kazuhito Ito

Graduate School of Science and Engineering, Saitama University
255 Shimookubo, Sakura-ku, Saitama, 338-8570, Japan
kazuhito@ees.saitama-u.ac.jp

Abstract— The Viterbi algorithm is widely used for decoding of the convolutional codes. To find the survivor path, the trace-back method is often employed because it consumes less power than the register exchange method especially for convolutional codes with many states. The disadvantage of the conventional trace-back using decision bits is the long decode latency. In this paper, a method of trace-back with source states instead of decision bits is proposed which reduces the number of memory accesses. The dedicated memory is also presented which supports the proposed trace-back method. The reduced memory accesses result in smaller power consumption and shorter decode latency than the conventional method.

I. Introduction

The Viterbi algorithm is widely used for decoding of the convolutional codes [1, 2]. A Viterbi decoder (VD) receives the codes which are generated by an encoder and may have some error during the communication or the storage. Then the VD constructs a trellis diagram based on the received codes and identifies the path on the diagram which corresponds to the most likely state transitions of the encoder. When the path is identified, the original information is decoded as such an input sequence that made the encoder cause the state transitions along the path.

The trellis diagram and hence the length of the path grow as long as the encoding and decoding processes continue. From the limitation in the VD implementation, the path is truncated to a finite length called the survivor path length L. Usually, L is chosen as several times larger than the constraint length K.

In conventional VDs, finding the path with the maximum likelihood is done by either register exchange method (RX) [3, 4] or trace-back method (TB) [5, 6, 7]. For VDs with larger K and L [8, 9, 10], or in power aware designs [11], the TB is used. In the TB method, the necessary information about the trellis diagram is stored in the survivor memory and the information is read out from the memory in backward direction to identify the most likely path. The drawback of the TB is the long latency of decoding because it is necessary to read the memory many times. In addition, despite the smaller power consumption than the RX, the memory access is still a major part of the power consumption of the decoder.

In this paper, a TB method for the Viterbi decoding of rate-$1/n$ convolutional codes is proposed which uses source states instead of decision bits to identify the most likely state transitions. With the aid of a special memory, which is also presented in this paper, the proposed method reduces the number of memory accesses necessary for TB and the decode latency.

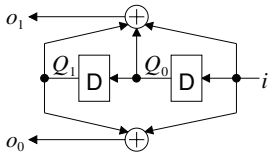

Fig. 1. An encoder for $K = 3$, rate-1/2 convolutional code.

This paper is organized as follows. In Sect. 2, the Viterbi decoding process is briefly reviewed. The proposed method of the TB with source states is presented in Sect. 3. The Viterbi decoder design using the proposed TB method is described in Sect. 4. The experimental results are presented in Sect. 5. Section 6 concludes this work.

II. Preliminary

A. Encoder for the convolutional codes

An encoder for the convolutional code with the constraint length $K = 3$ is shown in Fig. 1. In this figure, a box marked with 'D' is a one-bit register and '$\oplus$' means exclusive-OR (XOR) operation. Let N denote $K-1$. The encoder accepts an input bit i and stores the previously accepted N bits as $Q_{N-1}, Q_{N-2}, \ldots, Q_0$. If the encoder outputs n bits for every k input bits, it is said the *coding rate* is k/n, or simply rate-k/n. The encoder shown in Fig. 1 is rate-1/2, because it outputs two bits o_1 and o_0 ($n=2$) for an input bit ($k=1$). The set of outputs generated at the same time is called a *symbol*.

After generating the symbol, the stored previous inputs are shifted and the current input i is stored as Q_0. The set of values of $Q_{N-1}, Q_{N-2}, \ldots, Q_0$ is called a *state* of the encoder. Therefore, the encoder takes one of 2^N states at a time and transitions its state as it encodes the input bits.

B. Trellis diagram and ACS

The trellis diagram (TD) is the expansion of the state transition diagram of the encoder in time domain. Figure 2 shows

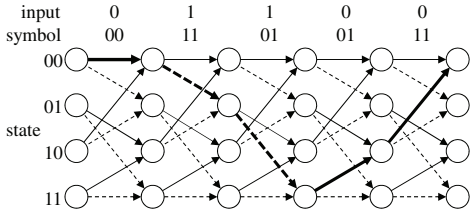

Fig. 2. The Trellis diagram for the encoder shown in Fig. 1.

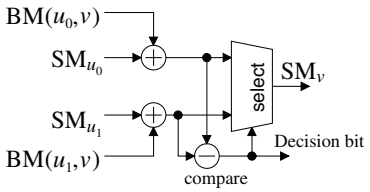

Fig. 3. The add-compare-select (ACS) unit.

the TD for the rate-1/2 convolutional code encoder in Fig. 1. In the TD, a directed arc (u,v) means a state transition from state u to v. If the transition is caused when the input bit is 0 (1), the arc is drawn by a solid (broken) line. Starting from the state 00, if the input bits are 0, 1, 1, 0, 0, the state transitions of the encoder follow the bold arcs in Fig. 2 and the output symbols are 00, 11, 01, 01, 11. For each state v, there are two possible state transitions (u_0,v) and (u_1,v) incoming to v. The decoder estimates which transition most likely occurred in the encoder. The likelihood is measured by *path metric* which is the sum of the *state metric* SM_{u_j} of the source state u_j $(j=\{0,1\})$ and the *branch metric* $\mathrm{BM}(u_j,v)$ of the state transition (u_j,v). The path with the smaller path metric is more likely and its metric is set as the state metric of v. The selected path is called the *survivor path*. The additions to calculate path metrics, compare, and select the smaller are performed by an add-compare-select (ACS) unit as shown in Fig. 3. The comparison result is called a *decision bit* (DB). For a received symbol, the ACS is done for each of the states. This process is called a *stage*.

C. Trace-back and decode

Given the state $(Q_{N-1},Q_{N-2},\ldots,Q_0)$ of the encoder, the last input bit to the encoder is obtained as Q_0.

In the trace-back (TB) method, 2^N decision bits for the current stage are stored at the write address WA in the memory called the *survivor memory*. The WA is incremented by 1 every stage. When sufficiently many stages are processed, the TB is performed as follows. Let $S=(s_{N-1},s_{N-2},\ldots,s_0)$, which is an N bit binary, denote the tentative state. The value of S is initialized to 0. By using the decision bit d of the ACS for state 0, S is updated as

$$(d,s_{N-1},s_{N-2},\ldots,s_1) \leftarrow (s_{N-1},s_{N-2},\ldots,s_0). \qquad (1)$$

The updated S gives the state in the 1st previous stage on the survivor path. Let the read address $\mathrm{RA}=\mathrm{WA}-1$. The memory data at RA are read and the bit d corresponding to S is selected. Then S is updated as Eq. (1). RA is decremented by 1 and the memory is read again. After repeating this $L-2$ times, S gives the most likely state of the encoder in the $(L-1)$-th previous stage. Once the most likely state S is identified, the input bit is decoded as s_0. Sometimes the above mentioned original TB is inefficient or unacceptable because $L-2$ memory read operations are required for every stage. Therefore, the periodic trace-back scheme is used in general [6, 9, 12]. In this scheme, decision bits are stored in the memory without the TB operation for L_1 stages. Then the TB is done to obtain the state in the L_2-th previous stage. The same trace-back operation is further repeated for L_1-1 times to decode the input bits for the (L_2+1)-th to the (L_2+L_1)-th previous stages. The former trace-back is called the *preliminary trace-back* (PTB) and the latter called the *decode* (DC). Hence the decoding of the input bits is performed every L_1 stages. In the periodic TB, the survivor path length varies from L_2+1 to L_2+L_1 depending on the stage. Because of the merging nature of the survivor paths, if L_2 is sufficiently large, S after the PTB gives the encoder state at the L_2-th previous stage very likely, whichever the starting S for the TB is chosen.

III. Trace-back with source states

Both the original and the periodic TB require many memory read operations. This results in the long decode latency as well as the large power consumption. The method to reduce the memory read operations is proposed here to improve the decode latency and the power consumption.

A. The conventional trace-back

Let Sv_p denote the state v in the stage p. An example of the DBs obtained for a trellis diagram is shown in Fig. 4(a). A bold arc (u,v) indicates the survivor path to the state v. The DB of 0 (1) for state v indicates that the survivor path to v comes through the upper (lower) arc among the two incoming arcs to v. For example, the survivor path to $S00_p$ is traced-back to $S00_{p-4}$ based on the DBs 1, 1, 0, and 0 which are read from the memory by four read operations.

The previous state is more explicitly given as shown in Fig. 4(b), where the previous state itself is read from the memory. For example, the value 10 for $S00_p$ directly indicates that the previous state of $S00_p$ is $S10_{p-1}$. The survivor path to $S00_p$ is traced-back to $S00_{p-4}$ based on the previous state information 10, 11, 01, and 00. The disadvantages of this method compared to using DBs are 1) more bits must be stored in the memory, 2) more memory bandwidth is required to store the previous state information, and 3) the number of memory read operations is not reduced.

B. The trace-back and decode with source states

The purpose of the PTB is to identify the state in the L_2-th previous stage on the survivor path. It is not necessary to obtain the intermediate states. Therefore, the above mentioned

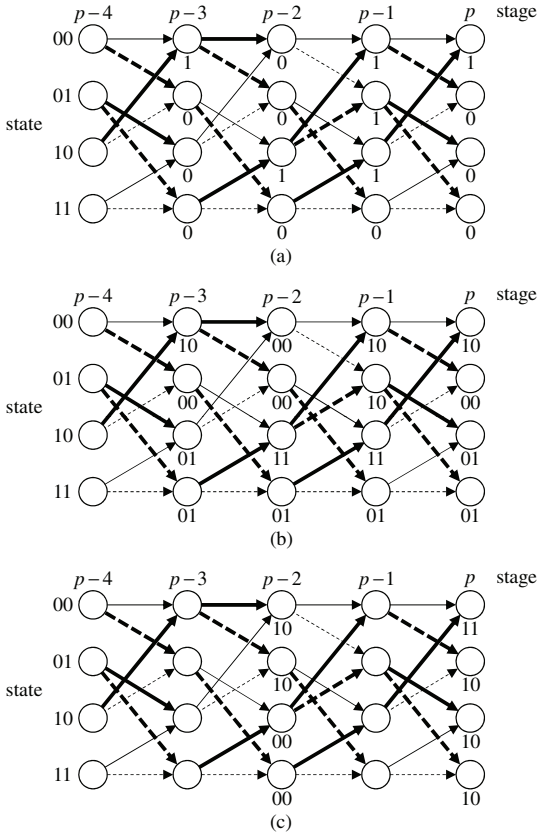

Fig. 4. The trellis diagrams for $K = 3$. (a) With decision bits. (b) With the previous state information. (c) With the source state information.

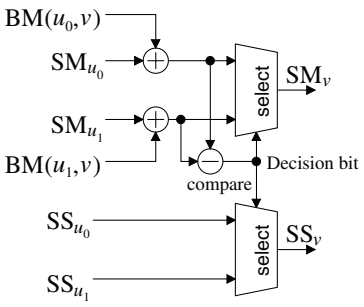

Fig. 5. The add-compare-select (ACS) unit for the TB with source state.

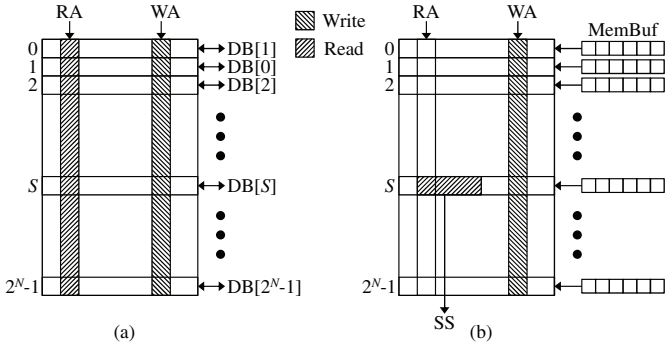

Fig. 6. The access of survivor memory. (a) For trace-back with decision bits. (b) For trace-back with source states.

method is modified such that not the previous but the 2nd previous state is read from the memory. This is illustrated in Fig. 4(c). For example, the value 11 for $S00_p$ indicates that the survivor path to $S00_p$ is traced-back to $S11_{p-2}$. The survivor path to $S00_p$ is traced-back to $S00_{p-4}$ based on the data 11 (for $S00_p$) and 00 (for $S11_{p-2}$), which are read from the memory by only two read operations.

The data stored in the memory are not necessarily the 2nd previous state information, but can be generalized to the m-th previous state information for $m \geq 2$. Let the information be called the *source state* and denoted as SSm. The state in the m-th previous stage on the survivor path to a state S is directly known as the SSm of S. The necessary number of memory read operations can be reduced to $1/m$ of the original TB.

The ACS unit needs to be modified as shown in Fig. 5 to compute the source state information. First, the source state of the state v, SS$_v$, is initialized to v. When the ACS decides that the survivor path to v comes from the state u_0 (u_1), SS$_{u_0}$ (SS$_{u_1}$) is selected as the new value of SS$_v$. After repeating this for m stages, SS$_v$ gives SSm of the state v. SS$_v$ is stored in the memory and initialized to v every m stages.

There exists a constraint on the value of m for the DC. If a state $S = (s_{N-1}, s_{N-2}, \ldots, s_0)$ in some stage is identified as the most likely state of the encoder, the last N input bits to the encoder are decoded as $s_{N-1}, s_{N-2}, \ldots, s_0$ as can be seen from Fig. 1. In order to decode the further previous input bits, the state in the N-th previous stage must be known. Therefore, m must not be larger than N to decode all the input bits.

C. The memory for the trace-back with source states

In the conventional TB with DB, 2^N DBs of all the states are written at the write address WA. The DBs at the read address RA are read from the memory and one of the bits is selected by the current tentative state S. The selected bit $d = \mathrm{DB}[S]$ is used to obtain the previous state according to Eq. (1). These are illustrated in Fig. 6(a).

In the TB with SS, 2^N SSs are written to the memory every m stages. Each SS is stored in MemBuf, which is an N-bit buffer register, and one bit is read serially per stage and written to the memory. By this technique, the total of $N2^N$ bits, N bits for each of 2^N states, are written to the memory in N stages. For tracing-back, the N bits stored at the read address RA and the consecutive $N-1$ addresses, and indicated by the tentative state S are read from the memory as the SS. These are illustrated in Fig. 6(b).

An ordinary memory consists of locations each of which stores a W-bit word. In the case of the TB with DB, the memory of $W = 2^N$ is used as the survivor memory so that all the DBs in a stage are stored at the same location. Usually the address is divided into the row and the column addresses. The storage for each bit of the word is arranged in a two dimensional array and a bit is stored to or read from the location specified by the row and the column addresses as shown in

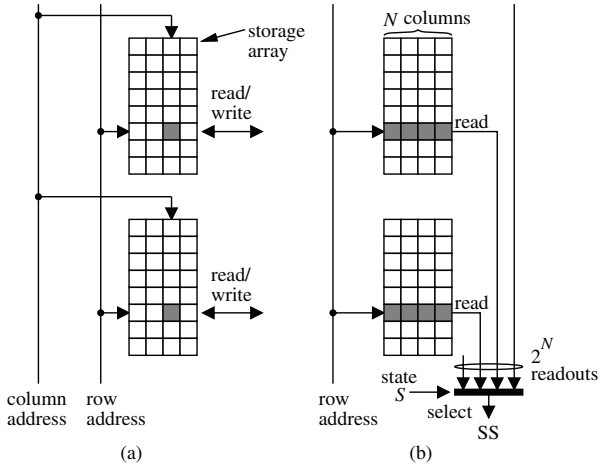

Fig. 7. The detailed architectures of survivor memories. (a) For trace-back with decision bits. (b) For trace-back with source states.

Fig. 7(a). The column address consists of c bits and generally ranges from 0 to 2^c-1. Because W arrays are used in the W-bit word memory, W bits are written to or read from the memory at the same time.

In the case of the memory for SSs, there are also $W=2^N$ storage arrays in the memory. The storage array consists of N columns and a sufficient number of rows. The address is divided so that the number of column address bits is $\lceil \log_2 N \rceil$ (N is not always a power of 2). The write operation is done in the same way as the ordinary memory as shown in Fig. 7(a). That is, a location to write a bit in each storage array is specified by both the row and the column addresses. The WA is incremented in such a way that the column address is incremented, and if the new value would be N, then it wraps around to 0 and the row address is incremented. Therefore, the N bits of an SS are stored in a row. In the read operation, all the bits stored on the row specified by the row address are read at the same time from every storage array. Then, one of them is chosen by the tentative state S as shown in Fig. 7(b).

IV. A Viterbi decoder design using the trace-back with source states

The proposed trace-back with source states (SS) and the memory design to support the SS are applied to implement an IS-95 Viterbi decoder [13]. The specification of the IS-95 Viterbi decoder is shown in Table I. Some parameters are adopted from the previous work [9]. The decoder is implemented in the same configuration as [9] where all the 256 state metrics, one for each of 256 states in a stage, are computed simultaneously by bit-serial ACS units. The bit-serial processing is used to reduce the hardware complexity.

A. Branch metric computation and path metric overflow

In the 3-bit soft decision, each received bit (with some error or noise) is quantized into 8 levels between $-\alpha$ and $+\alpha$ ($\alpha > 0$). Since the coding rate is 1/3, a symbol consists of 3 bits.

TABLE I
Specification of IS-95 Viterbi decoder

Constraint length	$K=9$ ($N=8$)
The number of states	$2^{K-1}=256$
Coding rate	1/3
Generator polynomials	557, 663, 711
Survivor path length	$L_1=24$, $L_2=48$
Decision level	3-bit soft decision
Path metric	8 bits

Let the quantized levels of the 3 bits of a symbol be denoted as I_2, I_1, and I_0. The branch metric $\mathrm{BM}(B)$ of the state transition generating the symbol $B=(b_2,b_1,b_0)$ is computed as

$$\mathrm{BM}(B)=\sum_{j=0,1,2}(\alpha\beta_j-I_j)^2 \tag{2}$$

where $\beta_j=1$ if $b_j=1$ and $\beta_j=-1$ if $b_j=0$ for $j=0,1,2$. Adding the constant $3\alpha^2-I_0^2-I_1^2-I_2^2$ and dividing by 2α do not change the inequality relations among branch metrics and hence among state metrics. Therefore, the branch metric $\mathrm{BM}(B)$ is redefined as

$$\mathrm{BM}(B)=\sum_{j=0,1,2}(\alpha-\beta_j I_j). \tag{3}$$

By choosing the value of α as 3.5, the 8-level $I_j+\alpha$ can be expressed as an integer ranging 0 to 7. The branch metric is computed easily by summing up $I_j+\alpha$ if $b_j=0$ or $7-(I_j+\alpha)$ if $b_j=1$ for $j=0,1,2$.

With the branch metric described above, the path metric grows monotonically. Therefore the overflow of the path metrics must be considered. The simulation results show that the maximum difference of the path metrics in every stage is 126. This is smaller than the half of 255, which is the largest possible value of an 8-bit binary. Thus, the technique in [14] is used where only the least 8 bits of the path metric is maintained even in the case of the overflow. The most significant bit (MSB) of the subtraction result between two 8-bit path metrics gives which is the smaller.

B. Bit-serial ACS unit and trace-back

The architecture of the bit-serial ACS unit is shown in Fig. 8. The block FA is a full adder, the box with 'D' is a 1-bit register, and MUX is a multiplexor. $\mathrm{P0}(v)$ and $\mathrm{P1}(v)$ denote the two previous states of the state v, $\mathrm{SM}_{\mathrm{P0}(v)}$ and $\mathrm{SM}_{\mathrm{P1}(v)}$ are the state metrics of these previous states, and BM_v^0 and BM_v^1 denote the branch metrics of the respective state transitions. The path metrics are computed and stored in the 8-bit depth first-in, first-out (FIFO) buffers, and the smaller is selected as SM_v, the state metric of state v. The additions for path metrics and the comparison (subtraction) are pipelined by the registers marked as 'P'. These pipeline registers, the carry registers (marked as 'C'), and the borrow register (marked as 'B') are clocked by the bit-serial clock CLK_S. Because the path metric is 8 bits long and pipelining is used, an ACS is done in 9 clock cycles of CLK_S. The register for DB (marked as 'DB') stores the

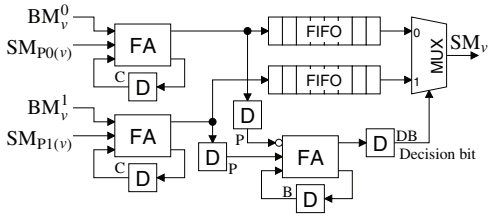

Fig. 8. The bit-serial ACS unit for trace-back with decision bits.

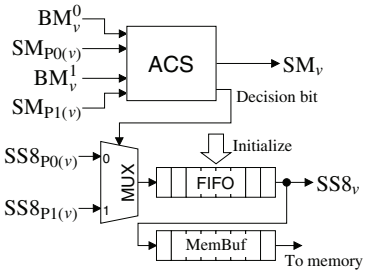

Fig. 9. The bit-serial ACS unit for trace-back with SS8.

comparison result at the 9th clock cycle, which is the MSB of the subtraction result.

The ACS unit for the TB with SS for $m = 8$ is shown in Fig. 9. In this figure, the block ACS corresponds to the bit-serial ACS unit shown in Fig. 8. The SS8 is selected based on the decision bit. The value of the SS8 is transferred to MemBuf and the SS8 is initialized to the binary expression of v every 8 stages. The MemBuf is also a FIFO and corresponds to the buffer shown in Fig. 6. By using MemBuf, the SS8 is written to the survivor memory one bit per stage. The timing chart of the operations of FIFOs is illustrated in Fig. 10.

The PTB and DC are done as follows. Let p denote the current stage. At first, the current value of the SS8 is obtained from the FIFO of the ACS of state 0. It gives the state in stage $p-8$. Then the SS8s in stages $p-8$, $p-16$, and so on are read from the survivor memory. Therefore, the PTB to stage $p-48$ ($L_2 = 48$) is done with 5 memory read operations. The 8 input bits for stages $p-56$ to $p-49$ are decoded as the binary expression of the traced-back state in stage $p-48$. To decode the remaining 16 input bits, the survivor memory is read 2 more times to trace-back to stages $p-56$ and $p-64$.

C. Decode latency

The detailed memory access timing is shown in Fig. 11. A stage consists of 9 clock cycles of CLK_S. The data are written to the memory in the 8th clock cycle. By the end of the 8th cycle in stage $24g$ (g is an integer), the SS8 is obtained in the FIFO. The value of the SS8 of state 0 gives the state in stage $24g-8$. It is accepted as an address by the memory at the rising edge of the memory clock CLK_M. After the memory access delay, the SS is output from the memory. It is denoted as SS8(1) in Fig. 11. The value SS8(1) is then used as the part of the memory address to read SS8(2). By repeating this,

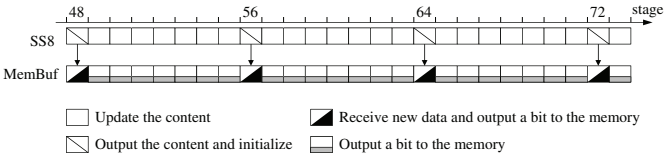

Fig. 10. The timing chart of operations of FIFOs.

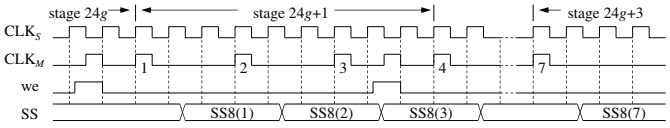

Fig. 11. The memory access timing for trace-back and decode.

SS8(5) is the traced-back state in stage $24g-48$. Two more data, SS8(6) and SS8(7), are necessary for DC. When the last symbol is processed in stage $24g-1$, the input bit in stage $24g-72$ is decoded in stage $24g+3$. If the decode latency is defined as the difference of stages between the receipt of a symbol and the decode of the input bit corresponding to the symbol, the decode latency is 75 stages.

V. Experimental results

The IS-95 Viterbi decoders with the proposed method were implemented using a 0.18μm CMOS process. The survivor memory is composed of 4 submemories so that the SSs are written to all the 4 submemories and a SS is read from one of them in TB. The submemories were designed as synchronous SRAMs. The power consumption is measured by Synopsys Prime Time for logic circuits with the back annotation and Synopsys Star-RCXT and Synopsys NanoSim for the survivor memory. Shown in Table II are the name of the design, the number of states of the encoder (# states), the number of transistors including the memory (# Tr.), the chip area in mm^2 (Area), the number of bits of the survivor memory (Mem), the achieved data rate (symbols per second) (DR), the survivor path length (SL), the decode latency in terms of the number of stages (Lt), the process used in μm (Proc), the power supply voltage in volts (VDD), the clock frequency in MHz (CLK), and the power consumption in mW (Power). In Table II, 'Conventional' is quite a similar design as presented in [9] which uses the TB with DBs. The differences from the design in [9] are 1) the same pipelined ACS unit shown in Fig. 8 is used and 2) the survivor memory is composed of 4 submemories. 'Proposed' is the design where the TB with SS8 is used.

The number of transistors and the power consumption in mW of the components used in the designed Viterbi decoders when operating at 900MHz clock are summarized in Table III. 'Core' includes 256 ACS units and the buffer trees for timing signals. 'TB' generates the address for the survivor memory. 'Timing' generates control signals for FIFOs in ACS units. 'BMU' computes the branch metrics. It is clear that the survivor memory is a major source of the power consumption of Viterbi decoders with the TB and the power consumption of the

TABLE II
SUMMARY OF THE IMPLEMENTED VITERBI DECODERS

Design	# states	# Tr.	Area	Mem	DR	SL	Lt	Proc	VDD	CLK	Power
[12]	64	—	0.54	16384	—	97 to 144	192	0.18	—	—	0.81*1
[10]	256	325k	5.76	—	20M	—	—	0.25	2.5	640	450
[9]	256	380k	10.25	24576	2M	49 to 72	96	0.5	1.8	16	9.8
Conventional	256	476k	1.325	24576	100M	49 to 72	96	0.18	1.8	900	505.5
Proposed	256	516k	1.450	16384	100M	49 to 72	75	0.18	1.8	900	431.8
Reg. Ex.	256	648k	1.074	0	100M	60	60	0.18	1.8	900	517.2

*1: expressed in mW/MHz

TABLE III
THE COMPONENTS USED IN THE DECODERS

Component	# Tr.	Power
Core (conventional)	181k	281.8
Core (Proposed)	283k	331.1
Memory (conventional)	290k	216.6
Memory (Proposed)	228k	95.86
TB (conventional)	2.52k	4.573
TB (Proposed)	1.57k	2.310
Timing (conventional)	1.53k	0.3755
Timing (Proposed)	1.53k	0.3755
BMU	1.15k	2.179

survivor memory is much reduced by the proposed method.

For comparison, the decoder with the register exchange (RX) was also implemented. The survivor path length of the RX is determined as 60 so that the coding gain is comparable to the TB. The implementation result is shown in the last row of Table II. Although the decode latency of the RX is the shortest, it consumes more power than the decoder with the TB when the same coding gain is achieved. The proposed TB with source states achieves exactly the same coding gains as the conventional TB with DBs, because the same ACS computations are executed and hence the same suvivor paths are obtained in the proposed and the conventional methods.

The proposed decoder consumes less power than the conventional decoder of the same constraint length, the survivor path length, and the data rate. The shorter decode latency and the lower power consumption than the conventional decoder are achieved by the proposed TB with SSs. In all the designs, the critical path is in the ACS unit. It is the multiplexor to select the state metric and the full adder to add the state metric and a branch metric.

VI. CONCLUSIONS

In this paper, a method of the TB which uses the source states is proposed. With the aid of the special survivor memory, which is presented in this paper, the number of memory accesses for the TB is reduced. This results in the lower power consumption and shorter decode latency than the conventional method. The TB with source states may well be used with high throughput Viterbi decoders [11, 15] where the look-ahead ACS is used to compute the state metrics from the 2nd or more previous states without computing state metrics of the intermediate states. The investigation of this would be an important task. In addition, the support for the punctured code with the coding rate k/n where $k > 1$ remains as future work.

ACKNOWLEDGEMENT

This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Inc. and Synopsys, Inc.

REFERENCES

[1] A.J. Viterbi, "Convolutional codes and their performance in communication systems," IEEE Trans. Commun., vol.19, no.5, pp.751–772, Oct. 1971.

[2] G.D. Forney Jr., "The Viterbi algorithm," Proc. IEEE, vol.61, no.3, pp.268–278, March 1973.

[3] P.J. Black and T.H.Y. Meng, "A 1-Gb/s, four-state, sliding block Viterbi decoder," IEEE Journal of Solid-State Circuits, vol.32, no.6, pp.797–805, June 1997.

[4] E. Yeo, S.A. Augsburger, W.R. Davis, and B. Nikolic, "A 500-Mb/s soft-output Viterbi decoder," IEEE Journal of Solid-State Circuits, vol.38, no.7, pp.1234–1241, July 2003.

[5] C.M. Rader, "Memory management in a Viterbi decoder," IEEE Trans. Commun., vol.29, no.9, pp.1399–1401, Sept. 1981.

[6] G. Feygin and P.G. Gulak, "Survivor sequence memory management in Viterbi decoders," Proceedings of IEEE Int. Symp. Circuits and Systems, pp.2967–2970, 1991.

[7] R. Cypher and C.B. Shung, "Generalized trace-back techniques for survivor memory management in the Viterbi algorithm," Journal of VLSI Signal Processing, vol.5, pp.85–94, 1993.

[8] I. Kang and A.N.W. Jr., "Low-power Viterbi decoder for CDMA mobile terminals," IEEE Journal of Solid-State Circuits, vol.33, no.3, pp.473–482, March 1998.

[9] Y.N. Chang, H. Suzuki, and K.K. Parhi, "A 2-MB/s 256-state 10-mW rate-1/3 Viterbi decoder," IEEE Journal of Solid-State Circuits, vol.35, no.6, pp.826–834, June 2000.

[10] X. Liu and M.C. Papaefthymiou, "Design of a high-throughput low-power IS95 Viterbi decoder," Proceedings of Design Automation Conference, pp.263–268, 2002.

[11] T. Gemmeke, M. Gansen, and T.G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," IEEE Journal of Solid-State Circuits, vol.37, no.7, pp.941–947, July 2002.

[12] S.C. Ma, "An improved one-pointer traceback method," IEICE Trans. Commun., vol.E87-B, no.4, pp.1016–1018, April 2004.

[13] Japan's Proposal for Candidate Radio Transmission Technology IMT-2000: W-CDMA, Association of Radio Industries and Businesses (ARIB), June 1998.

[14] H.L. Low, "Implementing the Viterbi algorithm," IEEE Signal Processing Magazine, vol.12, no.5, pp.42–52, 1995.

[15] J.J. Kong and K.K. Parhi, "Low-latency architectures for high-throughput rate Viterbi decoders," IEEE Trans. Very Large Scale Integration (VLSI) Systems, vol.12, no.6, pp.642–651, June 2004.