

An Efficient Pre-Traceback Approach for Viterbi Decoding in Wireless Communication

Yao Gang¹, Tughrul Arslan^{1,2}, Ahmet T Erdogan^{1,2}

¹School of Electrical Engineering, Edinburgh University, Edinburgh EH9 3JL, UK

²Institute for System Level Integration, Livingston, EH54 7EG, UK

Email: y.gang@ed.ac.uk

Abstract—An efficient *pre-traceback* architecture for the survivor path memory unit (SMU) of Viterbi Decoder (VD) targeting wireless communication applications is proposed. Compared to the conventional traceback approach which is based on three kinds of memory access operations: *decision bits write*, *traceback read* and *decode read*, the proposed architecture exploits the inherent parallelism between the *decision bit write* and the *decode traceback* operation by introducing a *pre-traceback* operation. The proposed *pre-traceback* approach reduces the survivor memory read operations by 50%. As a result of the reduction of the memory access operations, the size of the survivor memory as well as the decoding latency is reduced by as much as 25%. Implementation results show that the *pre-traceback* approach achieves up to 11% energy efficiency and 21% area saving compared to the conventional traceback architecture for typical wireless applications.

I. INTRODUCTION

Viterbi Algorithm (VA) with high constraint length K is a widely used error detection / correction scheme such as GSM, the voice channels of 3G and *IEEE 802.11a* wireless LAN. The well-known VA has been described in literature extensively [1] [2] [3]. The data path of the VD is composed of three major components: Branch Metric Calculation Unit (BMU), Add Compare Select (ACS) and Survivor Memory Unit (SMU).

ACS attracts most of the research efforts in the field of VD since the existence of feedback loop of the butterfly-like processor makes ACS unit the bottleneck of speed of the whole VD system [2] [3] [4]. However, in [5], it has been shown that SMU contributes over half of the power consumption and energy due to the extensive large memory access operations. Two basic approaches used to record survivor paths are register exchange and traceback in SMU of VD [3]. Up to now, only VDs with $K \leq 5$ implemented using register exchange have been reported in literature. In this paper, we focus on the traceback approach implementing SMU of the VD in high constraint length wireless applications. The techniques in previous work [5] [6] come at an expense of the decoding degradation to some extent.

The remainder of the paper is organized as follows: Section II briefly describes the conventional traceback. Section III deals with the proposed *pre-traceback* in which the algorithm as well as the implementation architecture is presented. Moreover, an example is also illustrated. Finally, in section IV the synthesis implementation results are examined to verify the analysis models and proposed architecture.

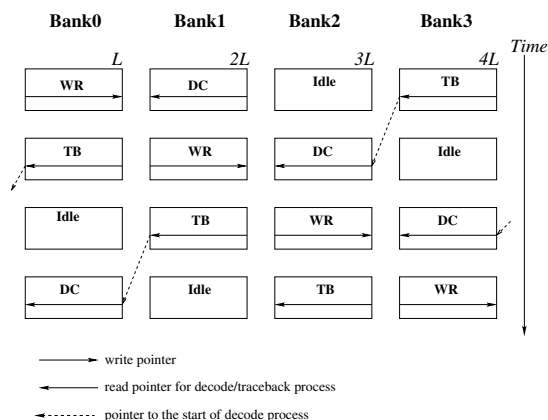


Fig. 1. Architecture Of Conventional Traceback Approach

II. CONVENTIONAL TRACEBACK

Generally, all the VDs with conventional traceback approach employ some variations of the k -pointer traceback architecture [7]. Here k refers to the number of the read pointers to access the survivor memory. The classical implementation of the trace back scheme is based on the property of unification [8]. That is, if we follow all survivor sequences back M stages, they all merge to the same state. Similarly, if we choose an arbitrary state at stage X and trace back to $X - M \gg L$ stages, we will reach the same maximum likelihood correct state.

The survivor path memory can be regarded as a circular 4 blocks of memory with L columns each as shown in Fig 1. Three operations *decision-vector write*, *traceback-read* and *decode-read* work in parallel to manipulate the decision bits vectors in different memory banks:

- *decision bit-write (WR)* writes the updated decision bit vectors from ACS unit into the survivor memory in an increased memory address order.
- *traceback-read (TB)* recursively estimates the previous state S_{n-1} according to the current state S_n and the associated decision bit d_n^s . For radix-2 applications, the estimation rule can be described as follows:

$$S_{n-1} = \{d_n^s, S_n \gg 1\} \quad (1)$$

The initial state is arbitrary according to the unification

property and the recursion will be repeated for consecutive L iterations.

- *decode-read (DC)* performs the similar read operation as the *traceback* operation defined by equation 1 except that the initial state is the output state estimated by *TB* in the last L time slot.

Every L time interval, the *TB* begins the trace back read from an arbitrary state, and the *DC* starts with the state determined by the *TB* process in the last L time interval. Since the decoded bits generated by the *DC* is in the reverse order, a simple two-stack Last In First Out (LIFO) scheme is used to perform bits order reversal. Each stack with a size of L depth, *DC* process pushes the reversal decoded bits stream into one stack while the decoded bits stored in the other stack are popped. Hence, the overall latency of traceback method, which is the time delay between the writing of the decision bit of *WR* process and the time when the corresponding bit is popped out of the LIFO, is $4L$. A key observation is that every decision bit in the survivor memory written by the *WR* will be read by *TB* and *DC* respectively. In other words, every decision bit in the survivor memory will be read two times. The redundant memory read operations are the main optimization target our proposed architecture try to address.

III. PROPOSED PRE TRACE BACK

A. Pre-Traceback Algorithm Description

The main task of the *TR* in conventional traceback approach is to get the initial start state for the *DC* through a recursive M shift and survivor memory read operations. For simplicity, we let the traceback length $M = L$. At time instant $L + M$, for $\forall i \in N = \{0, 1, 2, \dots, m\}$, where N is the set of trellis state and $m = 2^{K-1}$, S_{kL+M}^i will converge to the maximum likelihood state $S_{kL}^{max-like}$. Furthermore, during the time interval $[kL, (k+1)L]$, for $\forall i \in N$, we refer the pre state of i at the time instant kL as the *target trace back state*. Naturally, the goal of *TR* which starts at time instant $(k+1)L$ is to locate the *target traceback state* at time instant kL , which is the initial start state of the *DC* operation.

We propose an approach involving a *pre-traceback* operation through which the start of the *DC* can be obtained directly through a pointer register indicating the *target trace back state* instead of estimating the *DC* start state through a recursive *TR* operation. In our proposed approach, $\forall i \in N$, a pointer register which is denoted as S_n^i indicating the *target trace back state* of i at time instant n . At time instant n , in parallel with *WR* writes the decision bits into the survivor memory, the pointer register S_n^i is updated concurrently as follows:

$$S_n^i = S_{n-1}^{\{d_n^i, i \gg 1\}} \quad (2)$$

Here, d_n^i represents the decision bits of state i at time instant n . The exponential number, $\{d_n^i, i \gg 1\}$, represents the index number of the temporal adjacent pre state. The operation defined in equation 2 implies a multiplex select operation. It is apparent that the *pre-traceback* operation defined in equation 2 is performed in the forward direction opposed to the

conventional traceback operation which is done in backward direction. According to the unification property of the trellis graph, if L is set large enough, the pointer register S should point to the same state because of the emergence of the state. This fact is confirmed by our simulation. Hence, at time instant $(k+1)L$, for trellis state $\forall i \in N$, $S_{(k+1)L}^i$ should point to the maximum likelihood state:

$$S_{(k+1)L}^i = S_{kL}^{max-like} \quad (3)$$

which is the *target track back state* and the start state of the *DC*. As soon as *WR* complete $L + 1$ columns of decision bit update, the *target traceback state* of the last column is available at the same time. Hence, the corresponding *DC* can be performed from the *target traceback state* which is estimated by the *pre-traceback* operation directly. In our proposed *pre-traceback* approach, *pre-traceback* and *DC* are done in forward direction and backward direction respectively whereas in the conventional approach *TB* and *DC* are both done in backward direction. In summary, compared to the conventional traceback scheme which is based on three types of memory access operations (*WR*, *TB* and *DC*), only two types of operations are necessary in the *pre-traceback* approach, *WR* and *DC*:

- *decision bit-write (WR)* : In addition to update decision bit vectors in the survivor memory, the pointer register are also updated as described in equation 2. The update of the survivor memory and the pre-traceback of the pointer register are done in parallel.
- *decode-read (DC)* process performs two steps of operation:
 - First, select an arbitrary state and look for the *target traceback state* in the pointer register.
 - Second, from the start state, performs the iterative read operation for decoding.

B. Example

Further insight into the operations of the *pre-traceback* approach can be obtained through an example shown in Fig 2, where a four state trellis with the constraint length $K = 3$ is described. Each node corresponds to a state at a given time instant and each branch corresponds to a survivor path transition decided by ACS unit. The trellis state set N is indexed as $\{00, 01, 10, 11\}$. The corresponding decision bit associated with the survivor path transition is labelled under the node. For simplicity, a small time interval from kL to $kL+5$ is described, where $k = \{0, 1, 2, \dots\}$. The *WR* writes 5 columns of the decision bit vectors into the survivor memory in an increasing address order. In conventional traceback scheme, at time instant $kL+5$, in order to get the pre state of the 2(10) state at time instant kL , a series of shift operations specified as equation 1, are performed iteratively. At time instant $kL+5$, the decision bit of state 10 is 0, hence the pre state is 01. Similarly, the pre state of 01 is 00. After 5 consecutive memory read and shift operations, 10 is obtained, which is the pre state at time instant kL of the state 10 at time instant $kL+5$.

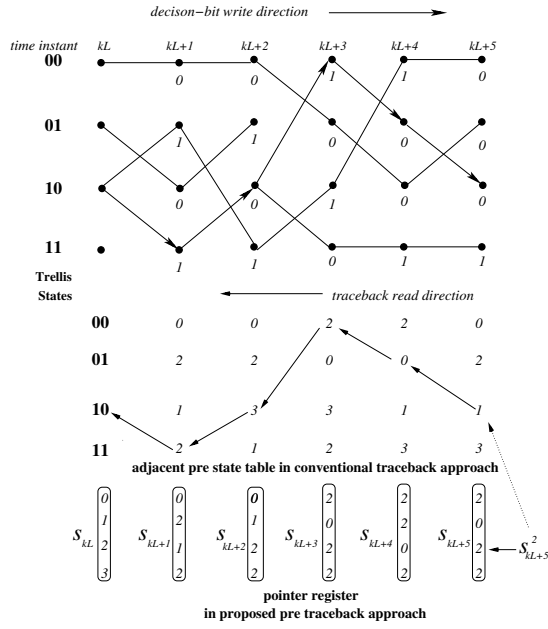


Fig. 2. Example: Comparisons Of Operations Of Conventional Traceback and Proposed Pre-Traceback Approach

The process of the traceback is in fact the process of building an adjacent pre state table. It can be seen that the traceback can only be done iteratively in that the conventional approach only maintains the information adjacently between time instant $kL + n$ and time instant $kL + n + 1$.

In our proposed *pre-rollback* approach, *TB* and *DC* are done in forward and backward direction respectively. During the *WR*, a pointer register which always points to the pre state of time instant kL of all the possible trellis state is employed. At time instant $kL + 1$, the pointer register is updated as $\{00, 10, 01, 10\}$, all of which point to the pre state at time instant kL . It should be noted it is the same as the pre state table as shown in the conventional approach in the sense that the two time instant are adjacent. Whereas at time instant $kL + 2$, the pointer register of state 10 denoted as S_{kL+2}^2 points to state 10 which is the target pre state at time instant kL compared to the adjacent pre state 11. Similarly, the whole column of register are updated as $\{00, 01, 10, 10\}$, which points to the pre state at time instant kL . Consequently, at time instant $kL + 5$, the pre state of the state 10, which is also state 10, can be obtained directly. Through the similar *pre-rollback* operation, at time instant $(k + 1)L$, the *target trace back* state of the state i , which is at time instant kL can be directly located instead of the recursive memory access operations in the conventional approach.

C. Architecture

The survivor memory is divided into three banks, with L columns for each bank, as shown in Fig 3. Only two types of operations: *WR* and *DC* work in parallel to access the survivor memory concurrently. In addition to the pointer register mentioned in the above section, another register is

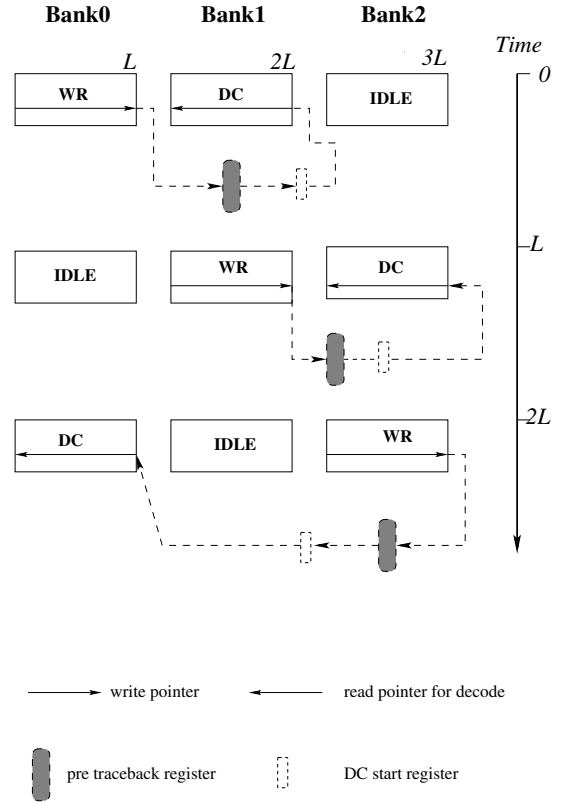


Fig. 3. Proposed Pre-Traceback Architecture

used to store the intermediate *DC* start state, which is simply referred to as *DC* start register. *WR* updates the decision vector columns in an increasing order. Correspondingly, the pointer register is updated based on the general description in equation 2. However, as we only need to know the *target traceback state* L stages before, the operation of *pre-rollback* is based on the block with the size of L . It should be noted that at time instant kL , where $k = \{1, 2, \dots\}$, *pre-rollback* for the time interval $[(k - 1)L, kL]$ and $[kL, (k + 1)L]$ is overlapped. A simple trick is employed with the aim to share the pointer register as well as overcoming the problem of overlap. It is apparent to observe that at time instant kL , for the *pre-rollback* operation at time interval $[kL, (k + 1)L]$, the i^{th} pointer register should always be initialized as i . So by just plugging the initial state number into equation 2 at time instant $kL + 1$, the pointer register between the two blocks can be shared. The *target traceback state* in the pointer register is shifted into *DC* start register at every kL time instant. Specifically, the operation of the pointer registers can be summarized as follows:

$$S_n^i = \begin{cases} i, & \text{if } n = 1 \\ i^{\{d_n^i, i \gg 1\}}, & \text{if } n = kL + 1, \text{ where } k = 1, 2, \dots \\ S_{n-1}^{\{d_n^i, i \gg 1\}}, & \text{otherwise} \end{cases} \quad (4)$$

At each kL time instant, with an initial start state determined by pointer registers, *DC* begins a consecutive memory read

TABLE I

SIZE AND LATENCY OF TRACEBACK AND PRE-TRACEBACK ARCHITECTURE

	Survivor Memory Depth	Decoding Latency
trace back	$4L$	$4L$
pre-traceback	$3L$	$3L$

operation with a decreasing address. A LIFO is also required to perform the bit reverse order. The overall latency including LIFO is only $3L$ opposed to $4L$ in the conventional approach. A simple architectural comparison is summarized in TABLE I. Compared to the conventional traceback approach, the proposed *pre-traceback* approach has 25% improvement in both memory size efficiency and latency reduction. Although a similar size and latency can also be achieved with the conventional traceback scheme according to [8], it must involve some form of dedicated clocking design strategies. Specifically, in [3], read pointer must work 3 times faster than write pointer and there is a stringent constraint on the phase relationship between the write pointer and the read pointer. This obviously improves the design efforts compared to the proposed *pre-traceback* approach which adopts a simple clocking strategy that the read and write pointers work at the same frequency.

IV. IMPLEMENTATION RESULTS

Two typical VDs in wireless communication field are implemented in UMC 0.18 μ standard cell environment:

- 3 bits, 8 level soft decision inputs
- code rates $R = 1/2$
- constraint length: $K = 7$ and $K = 9$, which corresponds to 64 and 256 states in the trellis graph

The BMU and ACS are exactly the same. For the convenience of the implementation, the truncation length L , is empirically set to 64. The proposed *pre-traceback* architecture takes advantages of the conventional architecture under the assumption of the overhead introduced by the pointer registers is negligible compared to the cost reduction by the survivor memory. The results presented in TABLE II summarizes the area synthesis results of the architecture specification of $K = 7$ and $K = 9$. For the typical wireless communication applications, i.e. wireless LAN where $K = 7$, the area savings in SMU block and the whole VD are 37.9% and 21.9% respectively.

10^4 random patterns were simulated at the clock frequency of 10 MHz for each VD, and the switching activities of each node were captured and then back annotated into the circuit. Power dissipation was then estimated for the synthesized gate-level circuits using Synopsys tools. and TABLE III, where $\hat{\eta}_{smu}$ represents the energy efficiency of the SMU unit and $\hat{\eta}_{vd}$ represents the energy efficiency of the whole VD. For typical applications in wireless LAN and 3G where $K = 7$ and $K = 9$ respectively, the energy efficiency of SMU is 22.8% and 21.6% respectively and the energy efficiency of the whole VD is 11.9% and 10.8% respectively.

TABLE II

SYNTHESIZED AREA RESULT: CONVENTIONAL TRACEBACK AND PROPOSED PRETRACEBACK ARCHITECTURE(mm^2)

K	7	9
SMU(Conventional Traceback)	2.24	8.98
SMU(Propose PreTraceback)	1.41	5.71
SMU Block Savings	37.9%	36.4%
VD(Conventional Traceback)	3.76	15.4
VD(Proposed PreTraceback)	2.94	12.1
VD Savings	21.9%	21.3%

TABLE III

ENERGY EFFICIENCY OF PROPOSED PRE-TRACEBACK APPROACH

K	7	9
$\hat{\eta}_{smu}$	22.8%	21.6%
$\hat{\eta}_{vd}$	11.9%	10.8%

V. CONCLUSION

An efficient *pre-traceback* architecture for the survivor memory unit for Viterbi Decoder (VD) targeting wireless applications is presented. The memory read operations are reduced by 50% by introducing the *pre-traceback* operation. The survivor memory size and the decoding latency is reduced by 25% compared to the conventional 2–pointer traceback algorithm. The combined reduction of survivor memory access operations as well as the survivor memory size leads to significant energy efficiency and area reduction.

REFERENCES

- [1] P. J.Black and T. H.Meng, "A 140-mb/s 32-state,radix-4 viterbi decoder," *IEEE Journal Of Solid-State Circuits*, vol. 27, p. 1877 to 1885, December 1992.
- [2] I.Kang and A.Willson, "Low power viterbi decoder for cdma mobile terminals," *IEEE Journal Of Solid-State Circuits*, vol. 33, p. 473 to 482, March 1998.
- [3] 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, p. 826 to 834, June 2000.
- [4] K.Parhi, "An improved pipelined msb-first add-compare select unit structure for viterbi decoders," *IEEE Transactions on Circuits and Systems I-Regular Papers*, vol. 51, p. 505 to 511, March 2004.
- [5] R. Henning and C. Chakrabarti, "An approach for adaptively approximating the viterbi algorithm to reduce power consumption while decoding convolutional codes," *IEEE Trans. On Signal Processing*, vol. 52, p. 1443 to 1451, May 2004.
- [6] Stanley.J.Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Transaction on Communications*, vol. 38, p. 3 to 12, January 1990.
- [7] G. Feygin and P.G.Gulak, "Architecture tradeoffs for survivor sequence memory management in viterbi decoders," *IEEE Trans. On Communications*, vol. 41, p. 425 to 429, March 1993.
- [8] R. Cypher and C. Shung, "Generalized trace back techniques for survivor memory management in the viterbi algorithm," in *IEEE Global Telecommunications Conference,GLOBECOM*, p. 1318 to 1322, December 1990.