

# An Efficient Pre-Traceback Architecture for the Viterbi Decoder Targeting Wireless Communication Applications

Yao Gang, Ahmet T. Erdogan, and Tughrul Arslan, *Member, IEEE*

**Abstract**—A large portion of silicon area and the energy consumed by the Viterbi decoder (VD) is dedicated to the survivor memory and the access operations associated with it. In this work, an efficient *pre-traceback* architecture for the survivor-path memory unit (SMU) of high constraint length 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 *decode traceback* operation by introducing *pre-traceback* operation. Consequently, the proposed *pre-traceback* approach reduces the survivor memory read operations by 50%. As a result of the reduction of the memory access operations, compared to the conventional 2-pointer traceback algorithm, 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* architecture achieves up to 11.9% energy efficiency and 21.3% area saving compared to the conventional traceback architecture for typical wireless applications.

**Index Terms**—Survivor-path memory unit (SMU), traceback, Viterbi decoder (VD).

## I. INTRODUCTION

THE most widely used technique for correcting errors in digital systems is the convolutional coding combined with a maximum likelihood (ML) decoding algorithm such as Viterbi algorithm (VA) [1]. This involves encoding the data with additional redundant information among the adjacent bits before transmission in which the width of correlated bits is referred to as constraint length  $K$ , then Viterbi decoder (VD) is used to find the most possible transmission bits to clean up the noise in received data. VA with high constraint length  $K$  is a widely used error correction scheme for communication systems such as global system for mobile communications (GSM), the voice channels of the third generation wireless communication technology (3G) and *IEEE 802.11a* wireless local area network (LAN).

The well-known VA has been described in literature extensively [2]–[5]. The data path of the VD is composed of three

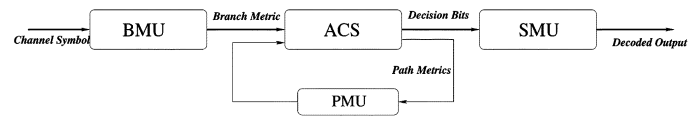


Fig. 1. Top level block diagram of a VD.

major components: branch metric unit (BMU), add compare select (ACS), and survivor-path memory unit (SMU) as shown in Fig. 1. The branch metrics are calculated from the received channel symbols in BMU and then fed into ACS which performs ACS for all the states. The decision bits generated in ACS are stored into and retrieved from the SMU in order to finally decode the source bits along the final survivor path. The path metrics of the current iteration are stored into the path metric unit (PMU) and read out for the use of next iteration.

ACS attracts most of the research efforts in the field of VD since the existence of the feedback loop makes it the bottleneck for the speed of the whole VD system [4]–[7]. However, the trend in the contemporary digital systems is that the memory unit is the primary concern of the designer in terms of the power consumption and area usage. The situation is even more critical for the VD design. In [8] and [9], it has been shown that SMU contributes over half of the power consumption due to the large number of memory access operations. This observation is also confirmed by various VD implementations [2]–[5].

In this paper, we focus on the SMU of the VD. In the literature, several schemes to reduce the memory access operations are available. A number of adaptive algorithms have been proposed in order to discard some survivor paths during the trellis recursion which lead to considerable energy and area savings in that only part of the trellis paths need to be stored and updated [9]–[12]. Systolic SMU hardware can be dynamically reconfigured to shorten the decoding depth when signal-to-noise ratio (SNR) increases [13], [14]. As a result of the shorter decoding depth, the survivor memory access operations are reduced too. However, the techniques mentioned above come at the expense of decoding performance degradation to some extent. In addition, when the channel SNR is low, the energy saving with these techniques is insignificant as there is a tradeoff between the decoding performance and power consumption.

One of the main contributions of this paper is to propose a novel *pre-traceback* architecture which reduces the memory read operations of the survivor memory by 50%. The *pre-traceback* architecture exploits the inherent parallelism between the *write* and *traceback* memory access operations. Furthermore, the size of the survivor memory as well as the decoding latency

Manuscript received September 16, 2004; revised September 5, 2005. This paper was recommended by Associate Editor Z. Wang.

Y. Gang was with the University of Edinburgh, Edinburgh EH9 3JL, U.K. He is now with Imagination Technologies, Hertfordshire WL4 8LZ, U.K. (e-mail: yao.gang@imgtec.com).

A. T. Erdogan and T. Arslan are with the University of Edinburgh, Edinburgh EH9 3JL, U.K. (e-mail: ahmet.erdogan@ee.ed.ac.uk).

Digital Object Identifier 10.1109/TCSI.2006.881187

is reduced by 25% with the proposed *pre-traceback* architecture. A brief synopsis of this work has been presented in [15].

The remainder of the paper is organized as follows. Section II briefly describes two classical architectures to decode the information out of the survivor paths: register exchange and conventional traceback [16]. 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. Section IV presents high-level estimation models for energy and area. Finally, in Section V the synthesis implementation results are provided in order to verify the efficiency of the proposed *pre-traceback* architecture and the high-level models.

## II. BACKGROUND

There are two basic approaches used to record survivor paths in VD. These are the register exchange and the traceback [1], [5], [17]. It has been traditionally argued that register exchange approach is proper for small constraint length VD and traceback is suitable for large constraint length VD. Due to the large constraint length of wireless communication applications, we focus on traceback approach in this paper.

### A. Register Exchange

Register exchange is the most direct method to extract the transmitted information from the encoded bits stream. Basically, the architecture consists of a two dimension register array between which is a column of multiplexers. The interconnection between the register arrays and the successive stages mimics the trellis graph. Register exchange is a good choice in terms of throughput because of its simple circuit structure and short critical path. However, for high constraint length applications in wireless communications, the wiring overhead in routing makes register exchange impractical[18]. Up to now, the work in the literature reveals that the register exchange is only implemented for VDs with  $K \leq 5$ .

### B. Conventional Traceback

Another method to extract the transmitted information from the encoded bits stream is based on the manipulation of the decision bits stored in the SMU. Generally, VDs with the conventional traceback approach employ some variations of the  $k$ -pointer traceback architecture [19]. Here  $k$  refers to the number of the read pointers to access the survivor memory. In [20], a generic traceback technique for survivor-path memory management is presented. By increasing the number of the read pointers [2] or the speed of the read pointer [5], the size of the survivor-path memory could be reduced. The main point is that there is a tradeoff between the total memory size, the latency and the design complexity. Hence, in the following, we start with the classical 2-pointer traceback algorithm and a more generalized discussion is presented in later sections.

The classical implementation of the traceback algorithm is based on the unification property [20]. That is, if we follow all survivor paths back  $M$  steps, they all merge to the same state. Similarly, if we choose an arbitrary state at time step  $X$  and traceback to  $X - M \gg L$  steps, we will reach the same ML state. As shown in Fig. 2, an arbitrary initial start state  $S_{L+M}^i$

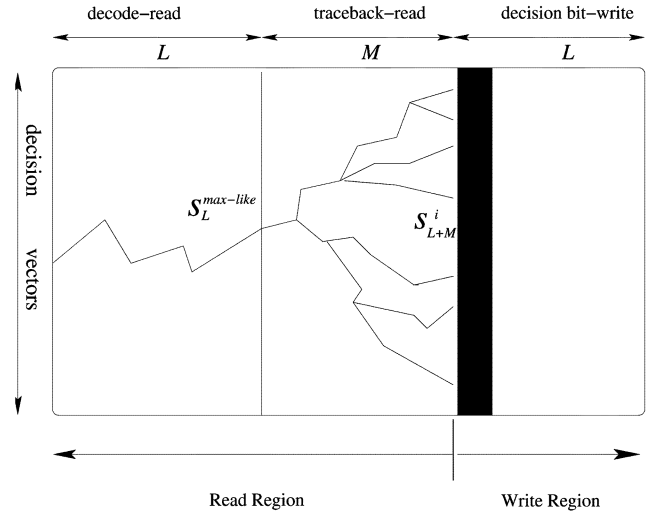


Fig. 2. Unification property of the traceback in Viterbi decoding.

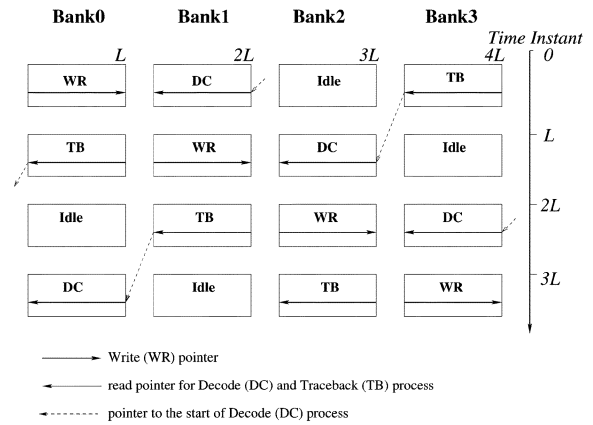


Fig. 3. Architecture of conventional traceback approach.

will converge to the ML state  $S_L^{\max\text{-like}}$  after a series of  $M$  step traceback iterations.

The survivor-path memory can be regarded as a circular four blocks of memory with  $L$  columns each as shown in Fig. 3. Three operations [*decision bit-write (WR)*, *traceback read (TB)*, and *decode read (DC)*] work in parallel to manipulate the decision bit vectors in different memory banks.

- *WR* writes the decision bit vectors provided by the ACS unit into the survivor memory in an increased memory address order.
- *TB* recursively estimates the previous state  $S_{n-1}$  along the path being traced back using the current state  $S_n$  and the associated decision bit  $d_n^s$  read from the survivor memory. The decision bit  $d_n^s$  is usually associated with the current state  $S_n$  by the index of the bit line of the memory. That is,  $d_n^s$  is located at the  $S_n$ th field of the bit line. The previous state  $S_{n-1}$  can be obtained as follows:

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

The function  $f(\cdot)$  is dependent on the structure of the trellis. For radix-2-based implementations, the above equation can be simplified into

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

Here,  $S_n \gg 1$  represents the operation of right shift  $S_n$  by 1 bit and the comma represents a concatenation operation between the two parts. In other words, the previous state is obtained by concatenating the decision bit and the right shifted current state by 1 bit. The initial state is selected arbitrarily based on the unification property and the recursion is repeated for  $M$  consecutive steps.

- DC performs read operations similar to the *traceback* operation defined by (2) except that the initial state is the output state estimated by TB in the last  $M$  steps. Furthermore, the decoded bits on the ML surviving path are retrieved in a reversing order by DC.

Every  $L$  time interval, TB begins the TB from an arbitrary state, and DC starts with the state determined by TB in the last  $M$  steps. Since the decoded bits generated by DC is in reverse order, a simple two-stack last-in-first-out (LIFO) scheme is used to perform bits order reversal. The overall latency of the *traceback*, which is the delay between the time when a decision bit is written into the memory by 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 WR is read by TB and DC, respectively. *In other words, every decision bit in the survivor memory is read twice. This redundancy in memory read operations is the main issue our proposed architecture addresses.*

### III. PROPOSED PRE-TRACEBACK

#### A. Pre-Traceback Algorithm Description

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

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

$$S_n^i = S_{n-1}^{\{d_n^i, i \gg 1\}}. \quad (3)$$

Here,  $d_n^i$  represents the decision bits of state  $i$  at time instant  $n$ , and  $\{d_n^i, i \gg 1\}$  represents the index for the pre-state. The operation defined in (3) implies a multiplex select operation as shown in Fig. 4. The  $i$ th pointer register  $S_n^i$  is updated according to the index  $\{d_n^i, i \gg 1\}$ . It is apparent that the *pre-traceback* operation defined in (3) is performed in the forward direction as

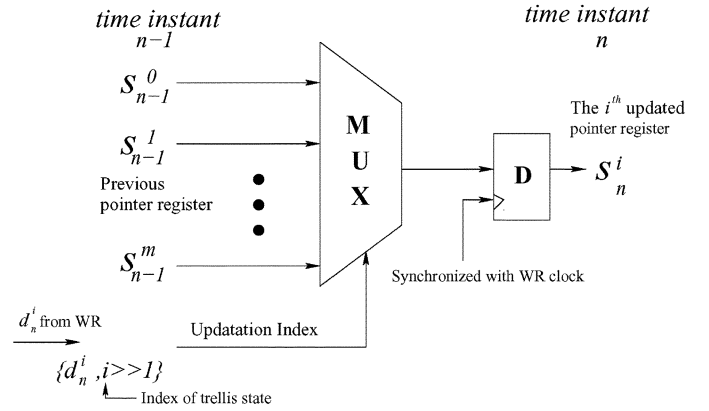


Fig. 4. Update operation of the proposed pre-traceback approach.

opposed to the conventional *traceback* operation which is performed in the 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 merge of the survivor paths. Hence, at time instant  $(k+1)L$ , for trellis state  $\forall i \in N$ ,  $S_{(k+1)L}^i$  should point to the ML state

$$S_{(k+1)L}^i = S_{kL}^{\max\text{-like}} \quad (4)$$

which is the *target traceback state* and the start state of DC. As soon as WR completes  $L + 1$  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 performed in forward and backward directions respectively. Whereas in the conventional approach TB and DC are both performed in the backward direction. In summary, compared to the conventional *traceback* approach which is based on three types of memory access operations (WR, TB, and DC), only two types of operations (WR and DC) are necessary with our *pre-traceback* approach.

- WR: In addition to updating the decision bit vectors in the survivor memory, the pointer register is also updated as described by (3). The update of the survivor memory and the *pre-traceback* pointer register are performed in parallel.
- DC: This process performs two operational steps.
  - First, selects an arbitrary state and looks for its corresponding *target traceback state* in the *pre-traceback* pointer register.
  - Second, starting with the state identified with the first step above, performs iterative read operations for decoding for the bits read from the memory.

The *pre-traceback* approach discussed in this paper can also be applied to similar ML decoding algorithms such as Turbo decoding which is implemented via a soft output VA (SOVA) decoder [18].

#### B. Example

Further insight into the operations of the *pre-traceback* can be obtained through an example as shown in Fig. 5, where a comparison between the conventional *traceback* and *pre-traceback*

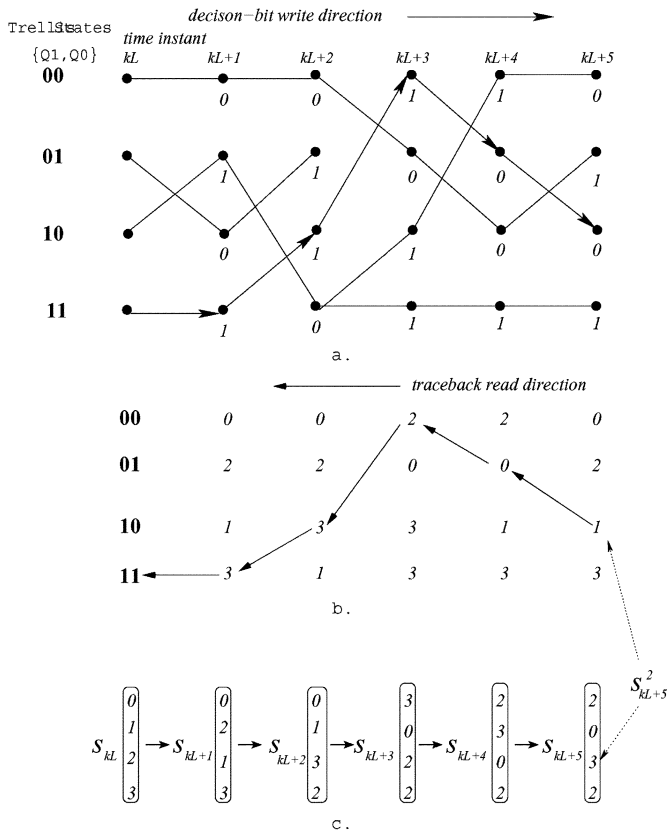


Fig. 5. Comparison of conventional traceback and the proposed pre-traceback approaches. (a) Trellis graph updated by WR process. (b) Conventional traceback through looking up adjacent pre-state table. (c) Pre-traceback through updating pointer register.

is provided using a four state trellis, corresponding to a constraint length  $K = 3$ . In Fig. 5(a), the trellis states are denoted as  $\{Q1, Q0\} = 00, 01, 10, 11$ . Each node corresponds to a state at a given time instant and each branch corresponds to a survivor path transition decided by the ACS unit. The corresponding decision bit associated with the survivor path transition is labelled under the node. For simplicity, a small time interval from  $kL$  to  $kL + 5$ , where  $k = \{0, 1, 2, \dots\}$  is considered without losing any generality. The WR process writes five columns of decision bit vectors into the survivor memory in an increasing address order.

Fig. 5(b) shows the conventional traceback scheme. For example, at time  $kL + 5$  if state 10 is selected as the start state, its corresponding pre-state at time  $kL$  can be obtained through a series of operations specified by (2). At time  $kL + 5$ , the decision bit of state 10 is 0, hence the corresponding pre-state at time  $kL + 4$  can be obtained as 01. Similarly, the pre-state of 01 is 00. After 5 such operations, state 11 ( $10 \Rightarrow 01 \Rightarrow 00 \Rightarrow 10 \Rightarrow 11 \Rightarrow 11$ ) is obtained as the target pre-state at time  $kL$  of the state 10 at time  $kL + 5$ . Therefore, the process of the traceback is in fact the process of building an adjacent pre-state table.

With the proposed *pre-traceback* approach, shown in Fig. 5(c), *pre-traceback* and DC are executed in forward and backward directions respectively. During WR, a pointer register which always points to the target pre-states at time  $kL$  of all possible trellis states is employed. At time  $kL$ , the pointer

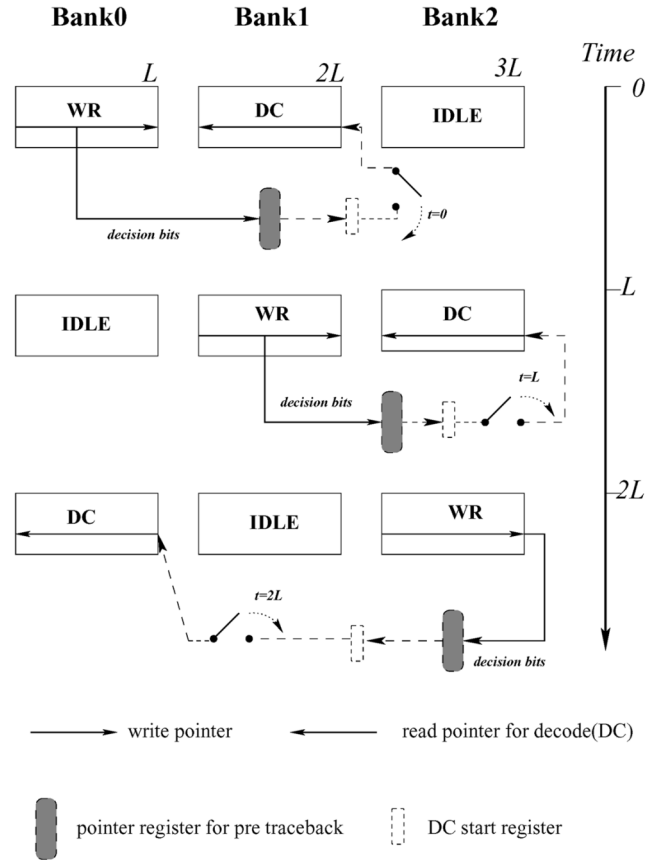


Fig. 6. Proposed pre-traceback architecture.

register is initialized to  $\{00, 01, 10, 11\}$ . At time  $kL + 1$ , the pointer register is updated as  $\{00, 10, 01, 11\}$ , all of which point to the pre-state at time  $kL$ . It should be noted that, for time  $kL + 1$ , the pointer register has the same states as with the pre-state table of the conventional approach, as shown in Fig. 5(b), since the time instants  $kL$  and  $kL + 1$  are adjacent. Whereas at time  $kL + 2$ , the pointer register of state 11 (denoted as  $S_{kL+2}^3$ ) points to state 10 (which is the target pre-state at time  $kL$ ) compared to the adjacent pre-state 01 at time  $kL + 1$ . Similarly, the whole column of pointer register is updated as  $\{00, 01, 11, 10\}$ , as specified by (3). Consequently, at time  $kL + 5$ , the target pre-state of state 10 (which is state 11) is directly available from the pointer register  $S_{kL+5}^2$ . Thus, with the *pre-traceback* approach the *target traceback state* of any given state can be directly obtained at the end of WR process, removing the need for recursive memory accesses required by TB process of the conventional traceback approach.

### C. Pre-Traceback Architecture

The survivor memory is divided into three banks, each with  $L$  columns, as shown in Fig. 6. Only two types of operations (WR and DC) are performed which work in parallel to access the survivor memory concurrently. In addition to the pointer register mentioned in the above section, another register is used to store the intermediate DC start state, which is simply referred to as DC start register. WR updates the decision bit vector columns in an increasing address order. Correspondingly, the pointer register is updated based on the general description in (3). It should

TABLE I  
SURVIVOR MEMORY SIZE AND DECODING LATENCY FOR TRACEBACK AND  
PRE-TRACEBACK ARCHITECTURES

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

be noted that at time  $kL$ , *pre-traceback* for the time interval  $[(k-1)L, kL]$  and  $[kL, (k+1)L]$  is overlapped. A simple trick is employed with the aim of sharing the pointer register as well as overcoming the problem of overlap. It can be observed that at time instant  $kL$ , for the *pre-traceback* operation at time interval  $[kL, (k+1)L]$ , the  $i$ th pointer register should always be initialized to  $i$ . So by just plugging the initial state number into (3) 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. The overhead introduced by the start register is negligible since all the pointer registers will converge to the same state. Hence, only one state is stored into the start register. Specifically, the updating process of the pointer register can be given as follows:

$$S_n^i = \begin{cases} i, & \text{if } n = 1 \\ I^{\{d_n^i, i \gg 1\}}, & \text{if } n = kL + 1 \\ S_{n-1}^{\{d_n^i, i \gg 1\}}, & \text{otherwise} \end{cases} \quad (5)$$

where  $I = \{0, 1, \dots, 2^{K-1}\}$ . At each  $kL$  time instant, with an initial start state determined by the pointer register, DC begins a set of  $L$  consecutive memory read operations with a decreasing address order. A LIFO is also required to perform the bit reverse order. Therefore, the overall latency including the LIFO is only  $3L$  as opposed to  $4L$  with 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 and latency reduction.

#### D. Generalized Architectural Comparisons Between Traceback and Pre-Traceback

The analysis above shows that the *pre-traceback* architecture is better than the traceback in terms of memory efficiency and decoding latency given that WR, DC and TB operate at the same clock frequency. It should be noted that for the traceback approach, the memory size and decoding latency can also be reduced by making the read bandwidth for TB and DC larger than the write bandwidth for WR according to [20]. In [19] and [20], a comprehensive theoretical analysis is presented for the architectures with different number of read and write pointers. Practical applications for these architectures are presented in [2], [4] and [5]. The main disadvantage of these architectures is increased design effort and complexity in synchronizing different pointers. Furthermore, some dedicated technologies such as multi port SRAMs are needed which reduce the flexibility

of the system. For example, in [5], the read pointer must operate three times faster than the write pointer and there is a stringent constraint on the phase relationship between the write pointer and the read pointer. These approaches increase the design complexity and design efforts compared to the proposed *pre-traceback* approach which adopts a simple clocking strategy [21]. In this subsection, we generalize the discussion to include the scenarios for the *pre-traceback* approach where WR and DC processes operate at different clock frequencies. We will show that the *pre-traceback* approach is still better than the conventional traceback approach when the read pointer bandwidth is increased as in [2] and [5]. Two detailed *pre-traceback* approaches are described in Figs. 7 and 8.

Fig. 7 illustrates the *2-pointer* *pre-traceback* approach where the read pointer for DC operates twice as fast as the write pointer for WR. The horizontal axis represents the time and the vertical axis represents the memory address. WR writes to the memory using a circular addressing scheme. The dots in the WR process represent the time when the target pre-state is made available for DC by the *pre-traceback* pointer register. Since the write throughput should be equal to the read throughput, the read pointer is not continuously active, as could be observed from Fig. 7. Compared to the architecture in Fig. 6, the memory size and the decoding latency is further reduced to  $2.5L$ . Similarly, if the read pointer works  $k$  times faster than the write pointer, the memory size and the decoding latency will be reduced to  $(2 + (1/k))L$ .

Furthermore, if two sets of pointer registers are allocated for each state, namely the even set and the odd set of pointer registers, the *pre-traceback* approach can be performed in parallel. With this arrangement, two sets of pointer registers operate concurrently at a time interval of  $L/2$ . The even set pointer register operates in the same way as the pointer register discussed before. It should be noted that at time instant  $L/2$ , the even set pointer register is updated as usual whereas the odd set pointer register is flushed to restart. Specifically, for the even set pointer register, the restart operation happens at time instants  $\{0, L, 2L, \dots, kL\}$  whereas for the odd set pointer register, the restart operation occurs at time instants  $\{L/2, 3L/2, 5L/2, \dots, (2k+1)L/2\}$ . Therefore, a new DC can start at each  $L/2$  time interval. Fig. 8 illustrates the detailed memory operation of this approach. Intuitively, we refer to this approach as the sliding-window approach. It should be noted that in this approach, the read pointer for DC is as fast as the write pointer for WR. Nevertheless, there are two *pre-traceback* processes operating in parallel, which corresponds to the “increased TB speed” in the conventional traceback approach. Therefore, for this architecture, the memory size and the decoding latency is further reduced to  $2L$  at the expense of doubling the overhead associated with the pointer register.

A comparison between the generalized traceback and *pre-traceback* approaches is presented in Table II.  $k$ -P-Even refers to the traceback approach where there are  $k$  read pointers for DC and TB. Whereas, One- $k$ -P refers to the traceback approach where there is a read pointer which works  $k$  times faster than the write pointer. The detailed descriptions of  $k$ -P-Even and One- $k$ -P approaches are presented in [19] and [20]. When  $k$  is small (e.g.,  $k = 1$  or  $k = 2$ ), the *pre-traceback* approach is

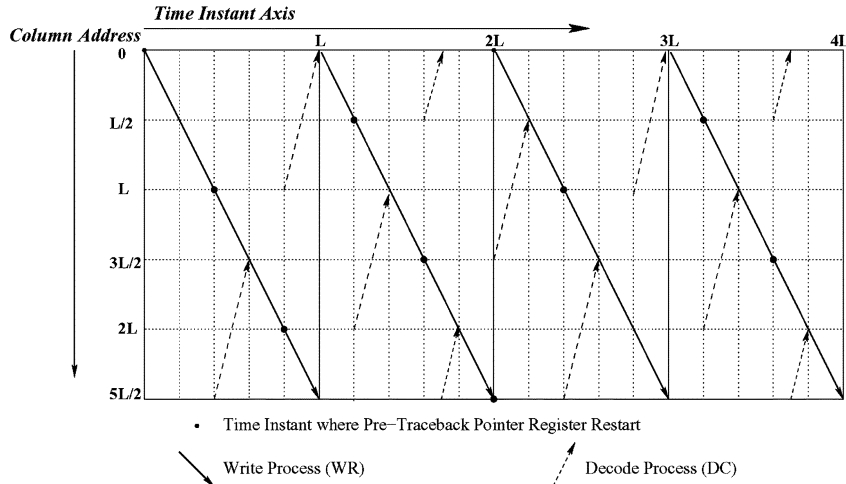


Fig. 7. Memory address partition for 2-pointer pre-traceback approach.

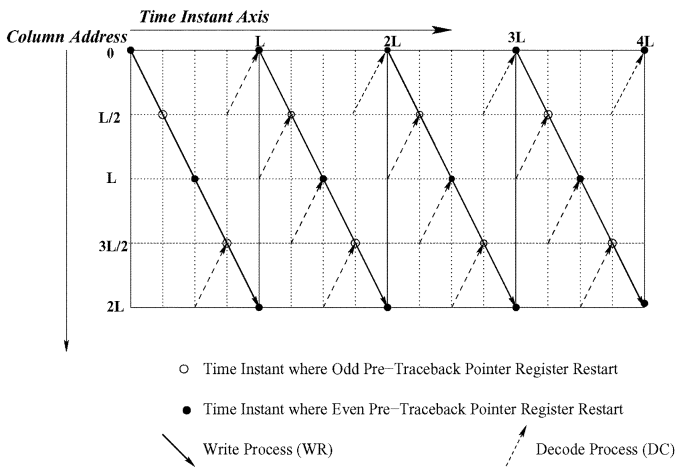


Fig. 8. Memory address partition for sliding window pre-traceback approach.

optimal compared to both  $k$ -P-Even and One- $k$ -P approaches. When  $k$  is large, One- $k$ -P begins to outperform the pre-traceback in terms of memory size. For example, for  $k = 3$  the memory requirement for  $k$ -P pre-traceback and One- $k$ -P trace-back is  $7L/3$  and  $2L$ , respectively. More vigorously, when  $k \rightarrow \infty$ , the corresponding memory depth for  $k$ -P pre-traceback and One- $k$ -P trace-back is approaching  $2L$  and  $L$  respectively. However, as it has been shown in [2], the additional complexity and overhead associated with simply increasing the read pointer bandwidth continuously will more than offset the gain achieved by reducing the memory size. Therefore, a practical range for  $k$  reported in the literature is up to 3.

Generally, the basic idea behind the traceback implementations is to reduce the memory size and decoding latency by squeezing the time consumed in the memory read process. However, from the power consumption point of view, each bit in the memory is still needs to be read twice, which leads to high memory access energy consumption. In summary, we can conclude that the pre-traceback is still optimal in terms of the overall design complexity, performance, and power consumption.

Finally, although the discussion presented above is conceptually interesting, the pre-traceback offers the most practical architecture for most applications where the read pointer operates at a modest speed. In the following sections, the analysis and experimental implementations are based on the architectural configurations shown in Figs. 3 and 6 where the read pointer operates at the same speed with the write pointer.

#### IV. SMU MODELING AND ANALYSIS

In this section, we present the first order approximation models for comparing the three approaches (register exchange, traceback, and pre-traceback). Although more accurate comparisons can be obtained only after implementation, it is always desirable to obtain an intuitive and quick understanding of the architectural factors affecting the overall system performance in terms of area, power, and speed [22]. The principal objective of modelling is to obtain some general comparisons between the conventional traceback and the proposed *pre traceback* architectures at the initial phase of the system design. Register exchange is also considered here for the sake of complete design space exploration. In addition, the models with generic architectural parameters are presented and certain approximations are made. In later sections, specific implementation results will be given to validate our models.

##### A. Energy Consumption

One of the main concerns in wireless communication applications is the energy consumption. In this subsection, the energy consumption models for the proposed *pre-traceback* and the conventional traceback approaches are presented respectively. The energy consumption of the traceback approach is contributed by the following operations:

- WR write operation into the survivor memory;
- TB read operation out of the survivor memory;
- DC read operation out of the survivor memory;
- write operation into LIFO;
- read operation out of LIFO;
- associated control logic.

TABLE II  
GENERALIZED MEMORY COMPARISONS BETWEEN PRE-TRACEBACK AND TRACEBACK

	Pre-Traceback		Traceback	
	$k$ -P	Sliding Window	$k$ -P-Even	One- $k$ -P
Memory Depth	$(2 + \frac{1}{k})L$	$2L$	$(2 + \frac{2}{k-1})L$	$(1 + \frac{2}{k-1})L$
Decoding Latency	$(2 + \frac{1}{k})L$	$2L$	$(2 + \frac{2}{k-1})L$	$(1 + \frac{2}{k-1})L$

In particular, for the conventional traceback approach, in order to decode one bit, the overall energy consumed can be modeled as

$$E_{\text{total}} = E_{\text{sram,wr}} + 2E_{\text{sram,rd}} + E_{\text{lifo,wr}} + E_{\text{lifo,rd}} + E_{\text{control}}. \quad (6)$$

Similarly, for the proposed *pre-traceback* approach, the energy consumption can be modeled as

$$E_{\text{total}} = E_{\text{sram,wr}} + E_{\text{sram,rd}} + E_{\text{lifo,wr}} + E_{\text{lifo,rd}} + \hat{E}_{\text{control}}. \quad (7)$$

Certain assumptions are made in order to get a quick overview of the power comparison between the two architectures. The energy consumed during the LIFO access between the two architectures is exactly the same. Moreover, as the size of the survivor memory is much larger than the LIFO, compared to the energy consumed during the survivor memory access, the energy consumed in the LIFO operation could be ignored. The introduction of the pointer register  $S$  results in  $\hat{E}_{\text{control}} \gg E_{\text{control}}$ . However, according to the results presented in [9], compared to the power consumption contributed by large SRAM access, the energy consumed in the control circuit could be ignored too. This assumption is further verified by our experimental implementation results in Section V. Hence, in the following analysis, the main contribution of the power comes from the survivor memory access operations. Therefore, the energy efficiency improvement of the proposed *pre-traceback* approach can be estimated as

$$\eta = 1 - \frac{E_{\text{sram,wr}} + E_{\text{sram,rd}}}{E_{\text{sram,wr}} + 2E_{\text{sram,rd}}} = 1 - \frac{1+r}{1+2r} \quad (8)$$

where  $r = E_{\text{sram,rd}}/E_{\text{sram,wr}}$ . Usually,  $r \leq 1$  since for read operations, a scheme to reduce voltage swing on the bit line is commonly employed in the contemporary SRAM Design [21]. The advanced techniques to derive  $r$  are introduced in [23], [24]. For example, with a  $K = 9$  VD, where the normalized  $E_{\text{sram,wr}}$  and  $E_{\text{sram,rd}}$  are 25 and 12 respectively which is extrapolated from [9], the proposed *pre-traceback* approach can achieve an energy efficiency improvement of  $\eta = 24.5\%$ . In this paper,  $r$  is simply considered as an empirical value derived from the simulation.

The energy efficiency  $\eta$  will vary with the variation of  $r$  which is a function of the memory size, row/column partition and technology. The generic relationship between  $r$  and energy

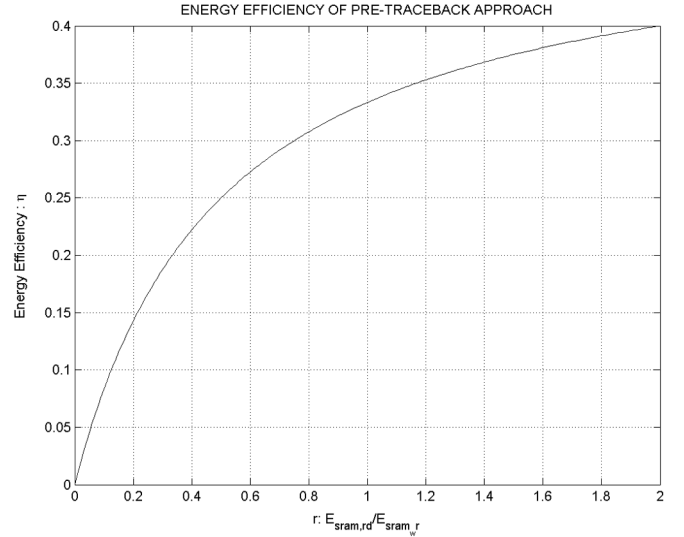


Fig. 9. Energy efficiency of the pre-traceback architecture.

efficiency  $\eta$  is presented in Fig. 9. It should be noted that in our model the energy consumption associated with the survivor memory access operations, for both conventional traceback and proposed pre-traceback approaches, are ideally modeled as equal. However, as the total memory size is reduced with the pre-traceback approach, the overall energy saving would be higher due to the reduced load capacitance. This will be confirmed later in the results section.

### B. Area

Similar to the power analysis, an approximation model is first presented and the synthesis implementation results are then provided in later sections in order to confirm the results. In [18], the area of a typical 6T memory cell in 0.18- $\mu\text{m}^2$  technology is given as  $10 \mu\text{m}^2$ . Address decoders and word/bit line drivers may increase this area by an overhead factor of 50%. Here the overhead factor, which is the area increase contributed by the peripheral circuits of SRAM such as address decoders and sense amplifiers, will vary according to the architecture and size of the SRAM. A typical  $D$  type register occupies about  $50 \mu\text{m}^2$ . The model parameters are listed as follows.

- $L$ : Decoding depth of the decoder.
- $N$ : Number of states in the trellis graph.
- $K$ : Constraint length, which determines the width of the pointer register array.
- $U_{\text{sram}}$ : Unit area of the SRAM cell.

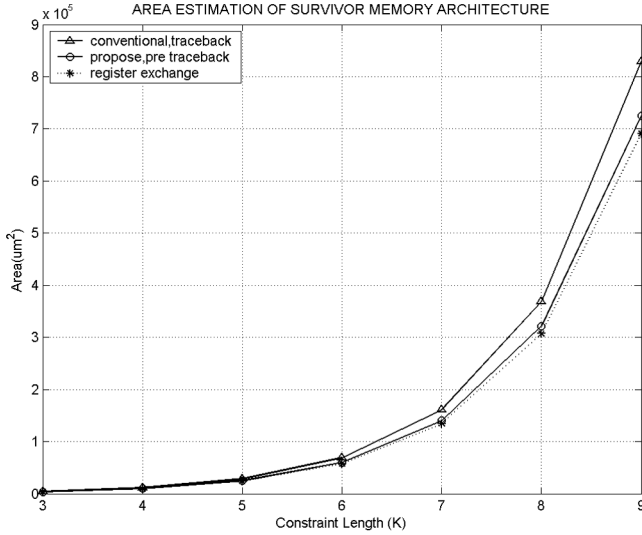


Fig. 10. Area estimation of traceback, pre-traceback and register exchange.

- $U_{\text{reg}}$  : Unit area of the register.
- $\alpha$ : Area overhead factor associated with the peripheral circuits such as decoder/sense amplifier of the SRAM.

For the conventional traceback approach, the area contribution comes mainly from the survivor memory, which can be modeled as

$$\text{AREA}_{\text{traceback}} = 4\alpha * N * L * U_{\text{sram}}. \quad (9)$$

For proposed *pre-traceback* approach, in addition to the survivor memory, the pointer register array should also be considered

$$\text{AREA}_{\text{pre-traceback}} = \underbrace{3\alpha * N * L * U_{\text{sram}}}_{\text{survivor\_memory}} + \underbrace{N * (K - 1) * U_{\text{reg}}}_{\text{pointer\_register}} \quad (10)$$

For the register exchange approach, it becomes more complicated to estimate the area in the sense that the wiring plays a major role in this architecture. Hence, the model shown in (11) just considers the ideal case where only the area of the register is taken into account. This provides a quick estimation for the area aspect of the register exchange approach

$$\text{AREA}_{\text{register exchange}} = N * L * U_{\text{reg}} \quad (11)$$

A generic relationship between the constraint length  $K$  and the area is shown in Fig. 10. It is obvious that the proposed *pre-traceback* approach is significantly more area efficient than the conventional traceback approach especially when the constraint length becomes larger. For example, for wireless LAN and 3G applications where  $K = 7$  and  $K = 9$  respectively, the proposed pre-traceback approach saves 22.6% and 23.1% area compared to the conventional traceback approach. Another interesting conclusion is that even when compared with the over

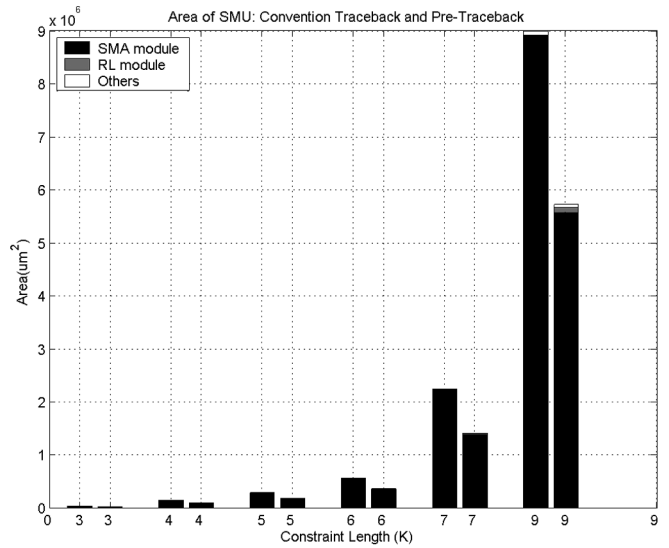


Fig. 11. Area contribution of traceback and pre-traceback.

ideal register exchange model, the area increase of the *pre-traceback* is not significant in that for  $K = 7$  and  $K = 9$ , the area increase of *pre-traceback* is only 4.3% and 4.8%, respectively.

## V. IMPLEMENTATION RESULTS AND DISCUSSION

In order to confirm the analysis shown above, several VDs with different SMU architectures are implemented in UMC 0.18- $\mu\text{m}$  standard cell-based environment. In these different VD implementations, BMU and ACS modules remain the same, the only difference being in their respective SMU modules. VDs are described in Verilog HDL and then synthesized into gate level circuits with Synopsys Design Compiler. The architectural parameters are as follows:

- 3 bits, 8 level soft decision inputs;
- code rate  $R = 1/2$ ;
- constraint length from  $K = 3$  to  $K = 9$ , which corresponds to 4 to 256 states in the trellis graph.

For the convenience of implementation, the decoding depth  $L$ , is empirically set to the number which is the power of 2 and no smaller than  $5K$ .

The proposed area models of the *pre-traceback* architecture in the previous section assumes that the overhead introduced by the pointer registers is negligible compared to the savings achieved with reduced survivor memory size. A careful examination of the logic circuit and survivor memory contribution to the SMU is presented here. The high level architecture of the SMU is composed of three major components: survivor memory array (SMA), recursion logic (RL) and LIFO. *RL* in *pre-traceback* architecture [defined by (5)] is much more complex than the *RL* of traceback [defined by (2)]. However, when compared to SMA, *RL* contributes little to the overall area in both conventional traceback and pre-traceback architectures. The increase is insignificant compared to the decrease in SMA for the pre-traceback architecture. Fig. 11 illustrates the area distribution of the SMU for both conventional traceback and proposed pre-traceback architectures. It should be noted that for each group of

TABLE III  
SYNTHESIZED AREA RESULT: CONVENTIONAL TRACEBACK AND PROPOSED PRE-TRACEBACK ARCHITECTURE IN MICROMETER SQUARE

K	3	4	5	6	7	9
L	16	32	32	32	64	64
SMU(Conventional Traceback)	43024	153746	294180	574591	2239849	8979621
SMU(Proposed PreTraceback)	29535	101266	190709	369925	1414038	5718978
SMU Block Savings	33.4%	34.6%	33.0%	36.6%	37.9%	36.4%
VD(Conventional Traceback)	71707	237026	508082	915647	3767926	15350405
VD(Proposed PreTraceback)	3.018	184546	404611	710981	2942115	12089762
VD Savings	19.2%	22.2%	21.4%	22.3%	21.9%	21.3%

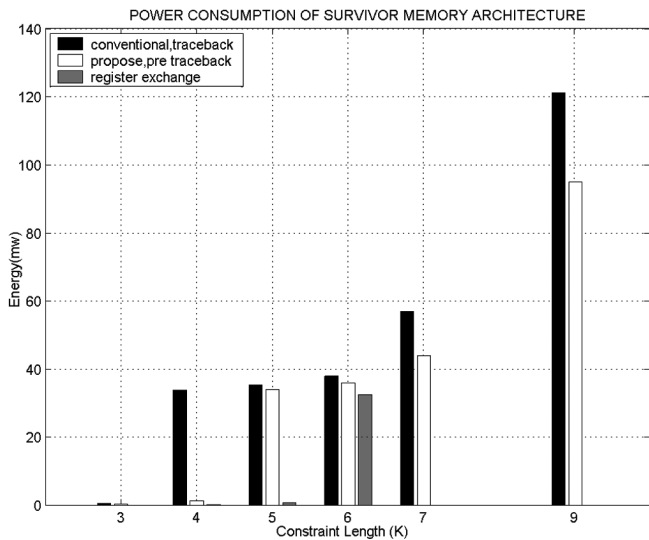


Fig. 12. Energy comparison of traceback, pretraceback and register exchange.

data with the same constraint length, the left one represents the conventional traceback approach and the right one represents the pre-traceback approach. Moreover, the results presented in Table III summarize the area synthesis results of the architecture for  $K$  from 3 to 9. For instance, for 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.

A total of  $10^4$  random patterns were simulated at a 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 consumption was then estimated for the synthesized gate-level circuits using Synopsys DesignPower. The energy comparison is shown in Fig. 12 and Table IV, where  $\hat{\eta}_{smu}$  represents the energy efficiency of the SMU and  $\hat{\eta}_{vd}$  represents the energy efficiency of the whole VD. For typical applications

TABLE IV  
ENERGY EFFICIENCY OF PROPOSED PRE-TRACEBACK APPROACH

K	3	4	5	6	7	9
$\hat{\eta}_{smu}$	30.1%	95.6%	4.5%	5.3%	22.8%	21.6%
$\hat{\eta}_{vd}$	15.1%	47.9%	2.4%	4.8%	11.9%	10.8%

in wireless LAN and 3G where  $K = 7$  and  $K = 9$ , respectively, the energy efficiency of the SMU is 22.8% and 21.6% respectively and the energy efficiency of the whole VD is 11.9% and 10.8%, respectively. For the purpose of completeness, the register exchange architecture with  $K \leq 6$  is also implemented. It can be seen that for small constraint length VD, the register exchange architecture is a good choice for the designer as the wiring overhead is low.

The implementation results also confirm the efficiency of the estimation models presented in Section IV. For example, when  $K = 9$ , the energy efficiency is 21.6%, which is quite close to the theoretical bound 24.5%, according to (8). In our ideal analysis model, the overhead associated with the pre-traceback logic has not been included, which may deteriorate the results for the pre-traceback architecture. Therefore, we contribute this optimal implementation result to the enhanced memory access efficiency for the pre-traceback architecture since for the model we assumed that the energy associated with the memory in traceback and pre-traceback was equal, which was impractical.

It is interesting to note that when  $K$  is increased from 3 to 4, the power consumed with the conventional architecture sharply increases whereas this happens with the *pre-traceback* architecture when  $K$  is increased from 4 to 5. Consequently, for  $K = 4$ , a significant 95.6% power saving in SMU is achieved. Also note that in the case of the *register exchange* this sharp increase in power consumption occurs when  $K$  is increased from 5 to 6.

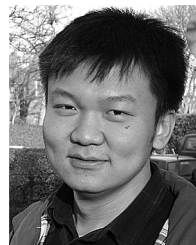
## VI. CONCLUSION

An efficient *pre-traceback* architecture for the SMU for VD targeting wireless applications is presented. The memory read operations are reduced by 50% by introducing the *pre-traceback* operation to exploit the inherent parallelism between WR and TB operations. Furthermore, the survivor memory size as well as 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. Results are presented for both the estimation models and experiment implementations. For typical wireless application fields such as *IEEE 802.11* wireless LAN ( $K = 7$ ) and 3G ( $K = 9$ ), the energy saving is 11.9% and 10.8% with a 21.9% and 21.3% area saving respectively.

## REFERENCES

- [1] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun.*, vol. COM-19, no. 10, pp. 751–771, Oct. 1971.
- [2] P. J. Black and T. H. Meng, "A 140-mb/s 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 6, pp. 1877–1885, Dec. 1992.
- [3] —, "A 1-Gb/s four-state, sliding block Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 797–805, Jun. 1997.
- [4] I. Kang and A. Willson, "Low-power Viterbi decoder for CDMA mobile terminals," *IEEE J. Solid-State Circuits*, vol. 33, no. 3, pp. 473–482, Mar. 1998.
- [5] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-state 10-mW rate 1/3 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 35, no. 6, pp. 826–834, Jun. 2000.
- [6] G. Fettweis and H. Meyer, "Parallel Viterbi algorithm implementation: Breaking the ACS bottleneck," *IEEE Trans. Commun.*, vol. 31, no. 10, pp. 785–790, Oct. 1989.
- [7] K. K. Parhi, "An improved pipelined MSB-first add-compare select unit structure for Viterbi decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 3, pp. 505–511, Mar. 2004.
- [8] D. Oh and S. Hwang, "Design of a Viterbi decoder with low power using minimum-transition traceback scheme," *Electron. Lett.*, vol. 32, pp. 2198–2199, Nov. 1996.
- [9] R. Henning and C. Chakrabarti, "An approach for adaptively approximating the Viterbi algorithm to reduce power consumption while decoding convolutional codes," *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1443–1451, May 2004.
- [10] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
- [11] F. Chan and D. Haccoun, "Adaptive Viterbi decoding of convolution codes over memoryless channels," *IEEE Trans. Commun.*, vol. 45, no. 11, pp. 1389–1400, Nov. 1997.
- [12] Y. Gang, T. Arslan, and A. T. Erdogan, "An efficient reformulation-based VLSI architecture for adaptive Viterbi decoding in wireless applications," in *Proc. IEEE Workshop Signal Process. Syst. (SIPS'04)*, Oct. 2004, pp. 206–210.
- [13] K. Takahashi, H. Tobita, S. Haruyama, and M. Nakagawa, "Reconfigurable systolic Viterbi decoder," in *Proc. Veh. Technol. Conf. (VTC'99)*, Sep. 19–22, 1999, vol. 3, pp. 1629–1632.
- [14] G. Fettweis and H. Meyer, "High rate Viterbi processors: A systolic array solution," *IEEE J. Select. Areas Commun.*, vol. 8, no. 10, pp. 1520–1534, Oct. 1990.
- [15] Y. Gang, T. Arslan, and A. T. Erdogan, "An efficient pre-traceback approach for Viterbi decoding in wireless communication," in *Proc. IEEE Int. Conf. Circuits Syst. (ISCAS'05)*, May 2005, pp. 5441–5444.
- [16] C. M. Rader, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. 29, no. 9, pp. 1399–1401, Sep. 1981.
- [17] D. A. El-dib and M. I. Elmasry, "Modified register exchange Viterbi decoder for low-power wireless communications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 2, pp. 371–378, Feb. 2004.
- [18] E. Yeo, S. Augsburger, Wm. R. Davis, and B. Nikolic, "Implementation of high throughput soft output Viterbi decoders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP'00)*, Jun. 2000, pp. 3378–3381.

- [19] G. Feygin and P. G. Gulak, "Architecture tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 425–429, Mar. 1993.
- [20] R. Cypher and C. Shung, "Generalized trace back techniques for survivor memory management in the Viterbi algorithm," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM'90)*, Dec. 1990, pp. 1318–1322.
- [21] J. M. Rabey, *Digital Integrated Circuits—A Design Perspective*, 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1996, pp. 417–419.
- [22] C. Chang, K. Kuusilinna, B. Richards, A. Chen, R. Brodersen, and B. Nikolic, "Rapid design and analysis of communication systems using the bee hardware emulation environment," in *Proc. IEEE Workshop Rapid Syst. Prototyping*, Jun. 2003, pp. 148–154.
- [23] R. J. Evans and P. D. Franzon, "Energy consumption modeling and optimization for SRAMs," *IEEE J. Solid-State Circuits*, vol. 30, no. 5, pp. 571–580, May 1995.
- [24] L. Yanbing and J. Henkel, "A framework for estimating and minimizing energy dissipation of embedded HW/SW systems," in *Proc. Design Autom. Conf. (DAC'98)*, Jun. 15–19, 1998, vol. 3, pp. 188–193.



**Yao Gang** received the B.S. degree in electrical engineering from Xi'an Jiao Tong University, Xi'an, China, and the M.S. degree (with distinction) from the University of Edinburgh, Edinburgh, U.K., in 2000 and 2005, respectively.

From 2000 to 2002, he was working in the ASIC Design Center, Huawei Technologies, Shenzhen, China, where he was a Design Engineer in charge of developing the broadband and optical transmission integrated circuits for the telecommunication equipment. From 2003 to 2005, he was with the School of Electrical and Engineering, the University of Edinburgh. He mainly focused on the low-power design of the wireless communication VLSI system. Since 2005, he has been working for Imagination Technologies, Hertfordshire, U.K., developing graphics intellectual property (IP) for communication and multimedia applications. His research interests include low-power VLSI design techniques and architectures for wireless communication and microprocessor.



**Ahmet T. Erdogan** received the B.Sc. degree in electronics engineering from Dokuz Eylul University, Izmir, Turkey, in 1990, and the M.Sc. and Ph.D. degrees from Cardiff University, Cardiff, U.K., in 1995 and 1999, respectively.

He is currently a Senior Research Fellow in System Level Integration in the School of Electronics and Engineering at the University of Edinburgh, Edinburgh, U.K. He is also a member of the Integrated Micro and Nano Systems Institute, University of Edinburgh and the Institute for System Level Integration, Livingston, U.K. His research interests include low-power VLSI Design, circuits and systems for communications and signal processing, design of energy-efficient digital circuits and systems, computer arithmetic, application-specific architecture design, and reconfigurable computing. He has published several journal and conference papers in these areas.



**Tughrul Arslan** (M'98) received the B.Eng. and Ph.D. degrees from the University of Hull, Hull, U.K., in 1989 and 1993, respectively.

He is a Professor in the School of Electronics and Engineering at the University of Edinburgh, Edinburgh, U.K. He is a member of the Integrated Micro and Nano Systems (IMNS) Institute and leads the System Level Integration group (SLI) in the university. He joined The University of Edinburgh from The Cardiff School of Engineering, where he directed the Advanced IC Design Prototyping Centre and the Intelligent Test Systems Group. His research interests include low-power design, DSP hardware design, system-on-chip (SoC) architectures, evolvable hardware, multiobjective optimization, and the use of genetic algorithms in hardware design issues.

Dr. Arslan was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING.