Parallel High-Throughput Limited Search Trellis Decoder VLSI Design

Fei Sun and Tong Zhang, *Member, IEEE*

***Abstract*—Limited search trellis decoding algorithms have great potentials of realizing low power due to their largely reduced computational complexity compared with the widely used Viterbi algorithm. However, because of the lack of operational parallelism and regularity in their original formulations, the limited search decoding algorithms have been traditionally ruled out for applications demanding very high throughput. We believe that, through appropriate algorithm and hardware architecture co-design, certain limited search trellis decoding algorithms can become serious competitors to the Viterbi algorithm for high-throughout applications. Focusing on the well-known T-algorithm, this paper presents techniques at the algorithm and VLSI architecture levels to design fully parallel T-algorithm limited search trellis decoders. We first develop a modified T-algorithm, called SPEC-T, to improve the algorithmic parallelism. Then, based on the conventional state-parallel register exchange Viterbi decoder, we develop a parallel SPEC-T decoder architecture that can effectively transform the reduced computational complexity at the algorithm level to the reduced switching activities in the hardware. We demonstrate the effectiveness of the SPEC-T design solution in the context of convolutional code decoding. Compared with state-parallel register exchange Viterbi decoders, the SPEC-T convolutional code decoders can achieve almost the same throughput and decoding performance, while realizing up to 56% power savings. For the first time, this work provides an approach to exploit the low power potential of the T-algorithm in very high throughput applications.**

***Index Terms*—Limited search trellis decoder, low power, parallel architecture, SPEC-T, T-algorithm, Viterbi algorithm (VA), VLSI.**

I. Introduction

TRELLIS decoding is pervasive in digital communication systems for error correction and signal detection. The well-known Viterbi algorithm (VA) [1] performs a breadth-first exhaustive search to realize maximum-likelihood (ML) trellis decoding. Because of its highly regular/parallel computation and data storage/retrieval operations, VA is well suited for high-throughput VLSI implementations and hence is being widely used in real-world applications. However, the use of exhaustive search makes the Viterbi decoder essentially not power efficient, particularly for applications demanding large trellises. To reduce the power consumption, we can use either reduced state sequence detection (RSSD) [2] or limited search trellis decoding [3]. RSSD applies the VA to a reduced trellis obtained by merging several states in the original trellis into one super state. For significant power reduction, RSSD typically suffers from large performance degradation, and how to reduce the trellis is nontrivial. In contrast, limited search trellis decoding algorithms perform limited (or nonexhaustive) search, as suggested by the name, on the original trellis. They have much less computational complexities than the VA, yet still achieve ML or near-ML performance. Intuitively, the largely reduced computational complexity may lead to great potentials of realizing low power.

Nevertheless, the real-world application of limited search trellis decoders pales in comparison to that of Viterbi decoder. This is mainly due to their lack of operational regularity and parallelism, which makes high-throughput hardware implementation problematic. Hardware decoder design has been studied for several limited search algorithms including Fano algorithm [4], stack algorithm [5], M-algorithm [6], and T-algorithm [7]. Since the Fano and stack algorithms are essentially path-serial, i.e., process only one path at one time, they are not suited for high-throughput applications but they can realize very low power consumption [8]. Most prior work [9]–[14] on these two algorithms focused on the data storage/retrieval structure design for moderate decoding speed-up. Belonging to the family of breadth-first search algorithms that also includes the VA, M- and T- algorithms perform nonexhaustive breadth-first search, where the number of survivor paths at each decoding depth is typically much less than the total number of trellis states, leading to much less computational complexities. Although M- and T- algorithms have greater potentials for parallel trellis decoding, all the previous work only targeted on the **path-serial** implementations of these two algorithms. Simons [15]–[17] developed sorting-based and nonsorting-based path-serial M- and T- algorithms decoders. Chan *et al.* [18] developed a path-serial T-algorithm decoder that is similar to the state-serial Viterbi Decoder but has longer critical path. We note that the drawback of irregular data storage/retrieval in M- and T-algorithms is completely *concealed* by the path-serial decoding process and thus is not an issue in path-serial decoders. The superior power efficiency of the T-algorithm has been recently demonstrated [19].

To the best of our knowledge, no limited search trellis decoder can achieve the throughput comparable to a state-parallel Viterbi decoder. In this paper, we propose an algorithm/architecture co-design solution to implement parallel a limited search trellis decoder that can realize significant power savings over its state-parallel Viterbi decoder counterpart for applications with large trellises, while achieving almost the same throughput and decoding performance. The underlying design methodology is the parallelism/regularity-driven algorithm/

Manuscript received August 5, 2004; revised May 22, 2005. This paper was presented in part at the IEEE International Symposium on Circuits and Systems (ISCAS), Vancouver, BC, Canada, May 23, 2004.

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: sunf@rpi.edu; tzhang@ecse.rpi.edu).

Digital Object Identifier 10.1109/TVLSI.2005.857181

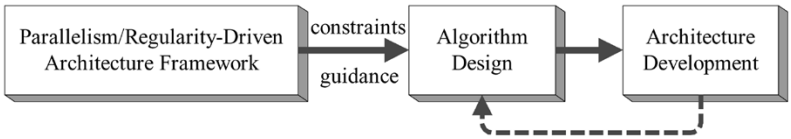


Fig. 1. Parallelism/regularity-driven algorithm/architecture co-design diagram (the dashed arrow represents the possible iterations between *Algorithm Design* and *Architecture Development*).

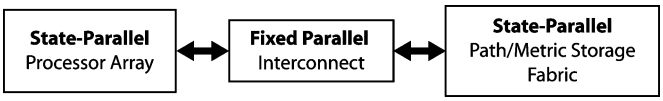


Fig. 2. Parallel decoder architecture framework.

architecture co-design as illustrated in Fig. 1. The first step in the co-design is to derive the principal decoder architecture framework. We note that a state-parallel Viterbi decoder has the abstract architecture framework as illustrated in Fig. 2, which has excellent operational parallelism/regularity. This motivates us to use this parallel architecture framework as the starting point of the co-design. Clearly, this architecture framework demands that the limited search algorithm explore trellis in a *state-parallel breadth-first fashion*. As breadth-first limited search algorithms, the M- and T- algorithms directly provide the groundwork for the algorithm design. To this end, there are two issues to be tackled, including the following.

1) *Decoding Speed-Up*: The main decoding data-paths of the conventional breadth-first limited search algorithms contain *serial* operations that prevent the decoder from achieving high throughput. Algorithm-level modification is required to eliminate such throughput bottleneck.
2) *Realization of Power Saving*: When a limited search algorithm is implemented on a parallel decoder, the largely reduced computational complexity at the algorithm level itself does not guarantee the realization of significant power saving. Appropriate architecture design should be developed to transform the reduced computational complexity to the reduced switching activities in the hardware for power saving.

This work chooses the T-algorithm as the basis for the algorithm design in the above algorithm/architecture co-design approach. First, we develop a modified T-algorithm, called SPEC-T, that fits to the state-parallel decoder architecture framework and eliminates the decoding throughput bottleneck in the original T-algorithm. Then we develop a parallel SPEC-T decoder architecture by modifying the conventional state-parallel register exchange (RE) Viterbi decoder architecture. The parallel SPEC-T decoder design involves a theme of trading silicon area for power savings: While it occupies larger silicon area than a state-parallel RE Viterbi decoder, it consumes less power because of the significantly reduced switching activities. The theme of *trading area for power savings* is well justified by the trend that power consumption other than the number of transistors is becoming the real limiter as CMOS technology continuously scales down.

To demonstrate the effectiveness of the proposed design solution, we designed parallel SPEC-T and corresponding state-parallel RE Viterbi decoders for rate-1/2 convolutional codes with 64, 128, and 256 states. The SYNOPSYS tool sets are used for synthesis and power estimation with 0.18 μm CMOS technology. Compared with the state-parallel RE Viterbi decoders, the SPEC-T decoders can realize almost the same throughput and decoding performance. The power savings of the SPEC-T decoders quickly increase as the number of states increases (e.g., almost no power savings at 64-state scenario but up to 56% power savings at 256-state scenario). Hence this design solution is more suitable to the applications demanding large trellises (such as CDMA IS-95 that uses a 256-state convolutional code). As the cost of power savings, the SPEC-T decoders occupy about 12% larger silicon area.

The remainder of this paper is organized as follows. Section II briefly reviews the background including the Viterbi Decoder and the original T-algorithm. The proposed SPEC-T algorithm and the parallel decoder architecture are presented in Sections III and IV, respectively. The design examples of SPEC-T and RE Viterbi convolutional code decoders are presented in Section V, and the conclusions are drawn in Section VI.

II. Background

A. Viterbi Decoder Basics

Since the parallel SPEC-T decoder hardware architecture is obtained by modifying the state-parallel RE Viterbi decoder, a brief review of Viterbi decoder is necessary. A Viterbi decoder mainly contains three functional blocks: 1) branch metric unit (BMU) that calculates all the branch metrics; 2) add-compare-select (ACS) units that update the accumulative survivor path metrics; and 3) survivor memory unit (SMU) that stores the survivor paths and generate the decoder output. For a trellis with N states, a state-parallel decoder implements all the N ACS units that operate in parallel.

As extensively discussed in the literature (e.g., [20]–[22]), SMU can be designed in two different styles, i.e., register exchange (RE) and trace back (TB), targeting on different power/area vs. throughput trade-offs. In general, RE can easily support very high decoding throughput but occupies larger silicon area and consumes more power; TB requires less silicon area and power but cannot support very high decoding throughput. In an RE Viterbi decoder, the decoder output is obtained by simple register shift operation and the critical path lies in the ACS recursion. On the other hand, in a TB Viterbi decoder, certain number of memory accesses are required to obtain each decoder output, which may make the trace back being the critical path. To support higher speed, TB will require complex memory structure design and may incur certain decoding performance degradation. An RE Viterbi decoder can generate output in two possible approaches: 1) output the last

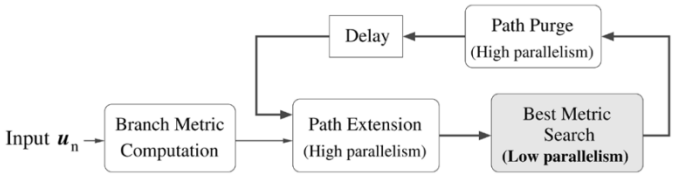


Fig. 3. Data-flow diagram of the T-algorithm.

(oldest) symbol of the survivor path led by a fixed trellis state or 2) apply a majority vote on the last symbols of all the survivor paths. To realize the same decoding performance, the latter approach requires a shorter decision length (i.e., the length of register exchange array) at the cost of implementing majority vote. Which one can achieve better silicon area and/or power consumption performance will depend on the trellis structure and specific hardware implementation styles.

B. Original T-Algorithm

The T-algorithm belongs to the family of breadth-first decoding algorithms. Broadly speaking, breadth-first decoding algorithms extend all the survivor paths at each trellis depth at once, purge some paths according to certain criterion, and then continue on to the next trellis depth. Various breadth-first algorithms primarily differ on the purging rules. Readers are referred to [3], [6] for more details. In the T-algorithm, at each decoding depth, all the paths whose cumulative path metric falls outside of a *retention band* will be purged. Its operations at each depth are outlined as follows with the data-flow diagram shown in Fig. 3.

1) *Branch Metric Computation and Path Extension*: Given input data $\mathbf{u}_n$ at depth n, compute the branch metrics and extend the survivor paths from the previous depth to obtain the contender paths at the present depth.
2) *Best Metric Search*: Find the contender path having the best (minimum) path metric, denoted as $\Gamma_B^{(n)}$, and release its oldest path symbol as the decoder output.
3) *Path Purge*: Purge the contender path whose metric $\Gamma^{(n)}$ satisfies $\Gamma^{(n)} - \Gamma_B^{(n)} > T$, where T is a pre-specified positive threshold.

The T-algorithm can achieve near-ML decoding performance with the average number of survivors much less than the total number of trellis states [7]. Hence it has great potentials of realizing low power. However, it is a challenge to implement a parallel high-throughput T-algorithm VLSI decoder mainly due to the following two reasons.

- *Algorithm-inherent path-parallel decoding throughput bottleneck:* As remarked in Fig. 3, although we can perform the *Path Extension* and *Path Purge* in parallel among all the paths with a short delay of only few additions and comparisons, the *Best Metric Search* incurs a relatively large delay due to the **serial** essence of a search operation, which prevents the decoder from achieving high throughput;
- *Irregular data storage/retrieval:* To enable parallel decoding, the decoder should be able to read and update all the survivor path data in parallel. However, the set of survivor paths varies from each decoding depth to the next. This makes the parallel path data access *dynamic* and *irregular*, whereas VLSI implementations always favor static and regular parallel data access such as that of a state-parallel Viterbi decoder. The irregular parallel data storage/retrieval may significantly degrade the throughput and increase power consumption of a parallel decoder.

III. Proposed SPEC-T Algorithm

Under the parallel decoder architecture framework as shown in Fig. 2, we developed a modified T-algorithm, called SPEC-T algorithm, that has two features. 1) It eliminates the algorithm-level throughput bottleneck due to the search-the-best-metric operation in the original T-algorithm. The basic idea is *best metric speculation with lagged correction*: instead of searching the exact best metric at each decoding depth, we *speculate* the best metric based on the current input and perform an *off-the-main-recursion* search to correct the speculation error with a certain delay. Therefore, we can move the serial search operation out from the main recursive decoding data-path and completely exploit the parallelism of the branch extension and path purge to speed up the decoding. 2) It works on the trellis structure of the signals while the original T-algorithm works on the tree structure of the signals (i.e., in the original T-algorithm each state may lead multiple survivor paths while in the SPEC-T each state can lead at most one survivor path). Fig. 4 shows the data-flow diagram of the SPEC-T algorithm. After the *Path Extension*, all the contender paths extended by the same trellis state go through a *Compare & Select* block that selects the winner and discard the other contender paths. The winner paths will be further processed by the *Path Purge* block that will select the final survivor paths based on the speculated best path metric. Given a decoder design parameter v that is a positive integer, the *Best Metric Speculation* block speculates the best path metric $\hat{\Gamma}_B^{(n)}$ as follows:

- if $n \bmod v \neq 0$, then $\hat{\Gamma}_B^{(n)} = \hat{\Gamma}_B^{(n-1)} + BM_B^{(n)}$, where $BM_B^{(n)}$ is the best branch metric given the input data. We note that the computation of $BM_B^{(n)}$ depends on the corresponding trellis coding scheme. In the context of convolutional code decoding, we may directly obtain $BM_B^{(n)}$ from the branch that matches the hard decision of the input $\mathbf{u}_n$;
- if $n \bmod v = 0$, then $\hat{\Gamma}_B^{(n)} = \hat{\Gamma}_B^{(n-1)} + BM_B^{(n)} + E$, where E is provided by the *Best Metric Search* block to compensate for the accumulated speculation error.

It is clear that we are very *optimistic* on the metric speculation, i.e., we always expect that the path with the best metric extends along the branch with the minimum branch metric. Clearly, this will introduce certain speculation error at each decoding depth, which will monotonically increase while continuing along the trellis. To prevent the accumulated speculation error from ever increasing, the SPEC-T algorithm *regularly* adjusts the depth-n speculated best path metric, for each $n \bmod v = 0$, based on the depth-$(n - v)$ accumulated speculation error provided by the *Best Metric Search*

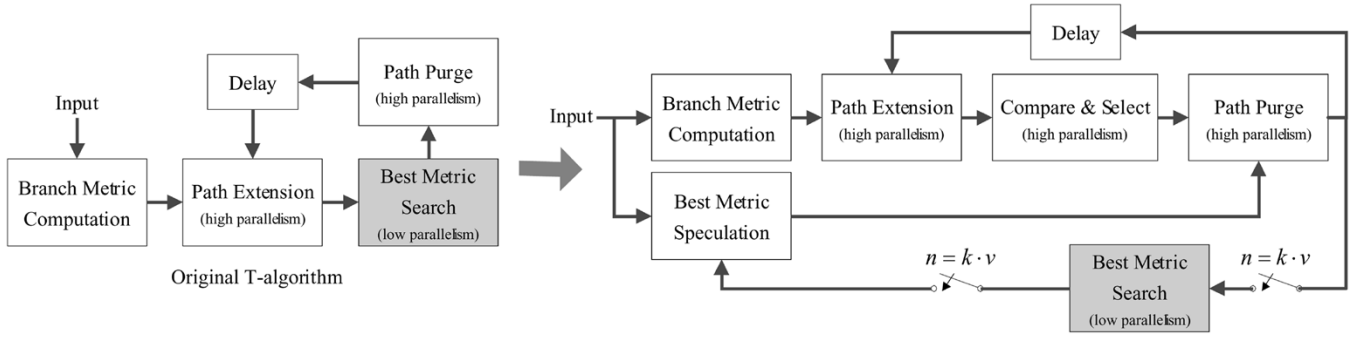


Fig. 4. Data-flow diagram of SPEC-T algorithm.

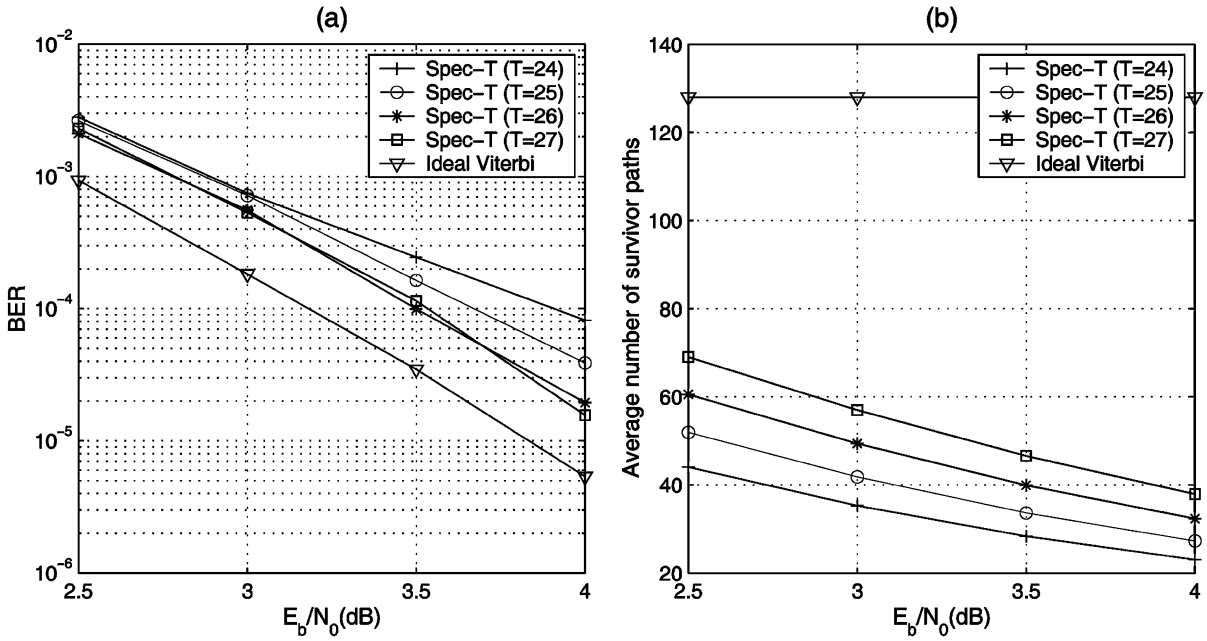


Fig. 5. Different threshold T values with $v = 7$ (rate-1/2 and 128-state). (a) BER versus SNR. (b) Average number of survivor paths versus SNR.

block. The depth-$(n - v)$ accumulated speculation error is $E = \min\left\{\Gamma_i^{(n-v)} - \hat{\Gamma}_B^{(n-v)}\right\}$, where $\Gamma_i^{(n-v)}$s are the metrics of the survivors at the $(n-v)$th depth. The *Best Metric Search* block finishes the search operation once every v depths. Hence the parameter v is called the speed mismatch factor between the main parallel recursive decoding data-path and the *Best Metric Search* block. We note that, when $v = 1$, SPEC-T algorithm reduces to the modified T-algorithm presented in [18].

By moving the serial search operation out from the main recursive decoding data-path, we can fully exploit the parallelism of the branch extension and path purge to speed up the decoding. To generate the decoding output, the SPEC-T algorithm can use two possible approaches that are similar to that of an RE Viterbi decoder: 1) output the last (oldest) symbol of one survivor path that is randomly chosen from the survivor paths at each depth or 2) apply a majority vote on the last symbols of all the survivor paths at each depth.

The decoding performance of the SPEC-T algorithm heavily depends on the threshold T and speed mismatch factor v. Meanwhile, both T and v will also affect the average number of survivor paths that directly determines the switching activities and hence power efficiency. In practice, the minimum value of v is determined by the specific implementations, e.g., the hardware design examples described in Section V have the minimum value of $v = 7$ (128 and 256 states). To illustrates their effect on the trade-off between the decoding performance and average number of survivor paths, let us consider the decoding of a rate-1/2 128-state convolutional code with the generator of (247, 371). We assume the signals are modulated by binary phase-shift keying (BPSK) and transmitted over an additive white Gaussian noise (AWGN) channel, and normalize the transmission power of each codeword bit normalized as 1 in the simulation. Fig. 5 shows the fixed point simulation results when we fix $v = 7$ and change the value of T, and Fig. 6 shows the simulation results when we fix $T = 26$ and change the value of v. For the purpose of comparison, we also show the results when ideal Viterbi algorithm (i.e., floating-point precision and infinite decision length) being used.

IV. Parallel SPEC-T Decoder Architecture

This section presents a parallel SPEC-T decoder hardware architecture obtained by modifying an RE Viterbi decoder. The main motivations of using RE Viterbi decoder as a design basis are: 1) a very high throughput decoding can be easily achieve

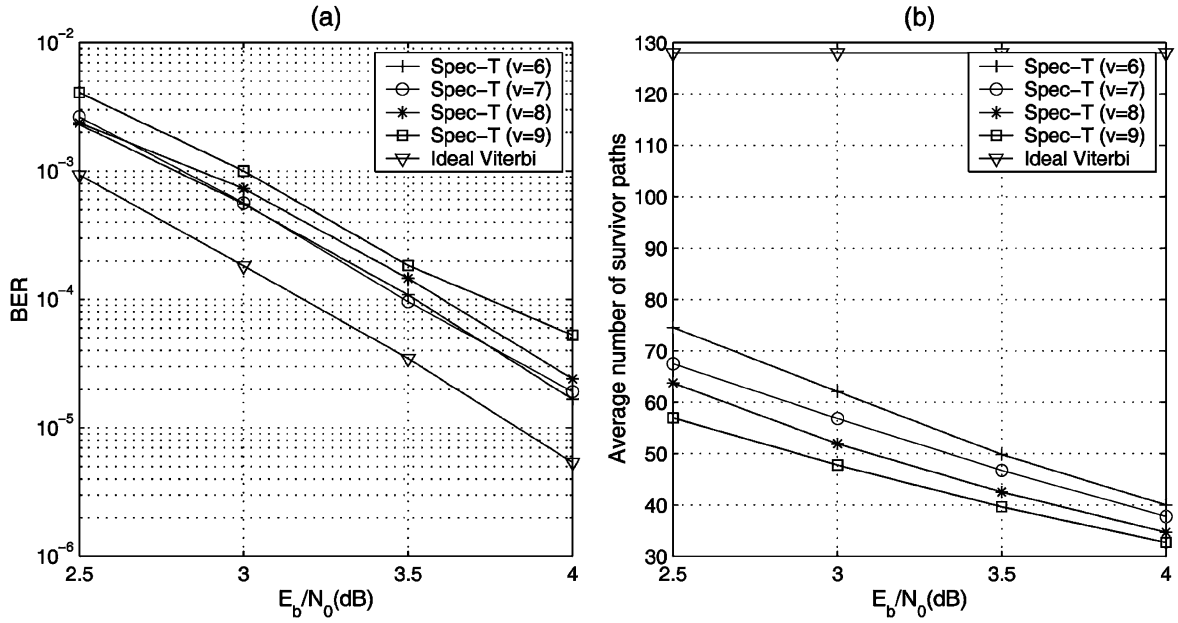


Fig. 6. Different mismatch factor v with $T = 26$ (rate-1/2 and 128-state). (a) BER versus SNR. (b) Average number of survivor paths versus SNR.

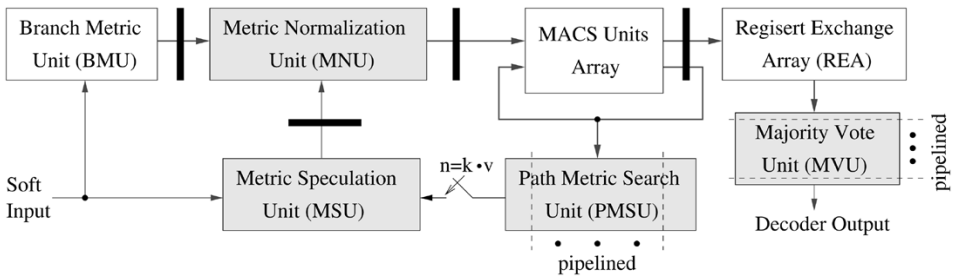


Fig. 7. Parallel SPEC-T decoder structure block diagram.

and 2) the use of register exchange structure makes it easy to leverage the reduced number of survivors to reduce the power consumption of the SMU.

A. Architecture Overview

The block diagram of the proposed parallel SPEC-T decoder is shown in Fig. 7. The shaded blocks are extra functional blocks added into the conventional state-parallel RE Viterbi decoder. The essential difference between the Viterbi decoder and SPEC-T decoder is that, in a Viterbi decoder, at each decoding depth, each trellis state generates one survivor path that participates the computation of the next decoding depth and invokes the corresponding register exchange operation; in a SPEC-T decoder, not all (usually only a small portion) of trellis states generates survivor paths. All the functional blocks are outlined as follows.

Branch Metric Unit (BMU): The BMU calculates the Euclidean distance between the input data and each distinct branch symbol of the trellis. In certain circumstances such as convolutional code decoding, the calculation can be largely simplified in order to reduce the silicon area and/or improve the speed, where the calculated branch metric is no longer the absolute Euclidean distance but will not (largely) affect the decoding performance [23].

Metric Speculation Unit (MSU): The task of MSU is to provide the speculated best (minimum) path metric. Denote the best branch metric at the nth decoding depth as $BM_B^{(n)}$, the threshold value as T, and the input from the path metric search unit as E. According to the above SPEC-T algorithm, we have: if $n \bmod v \neq 0$, MSU directly outputs $BM_B^{(n)}$; if $n \bmod v = 0$, MSU outputs $BM_B^{(n)} + T + E$. Notice that we use $BM_B^{(n)} + T + E$ instead of $BM_B^{(n)} + E$ in the above SPEC-T algorithm. This can eliminate the *compare-with-T* operation in the modified ACS units (as described later) and hence improve the decoding throughput because the decoder's critical path lies in the modified ACS units. In the context of convolutional code decoding, the best branch metric is the metric of the branch corresponding to the hard decision of the soft input.

Metric Normalization Unit (MNU): It normalizes the branch metrics by subtracting the output of the metric speculation unit from each branch metric. Then the normalized branch metrics are feed to the modified ACS units. The reason for introducing the normalization is explained as follows: Let w denote the output of the metric speculation unit. According to the above SPEC-T algorithm, w should be distributed to each trellis state and compared with the local winner, i.e., it will involve the calculation of $\Gamma_i + BM_j - w$, where Γ_i and BM_j denote the local winner metric and branch metric, respectively. This may lead to

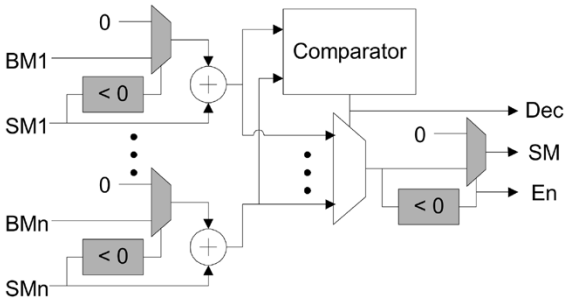


Fig. 8. The modified ACS unit in SPEC-T decoder.

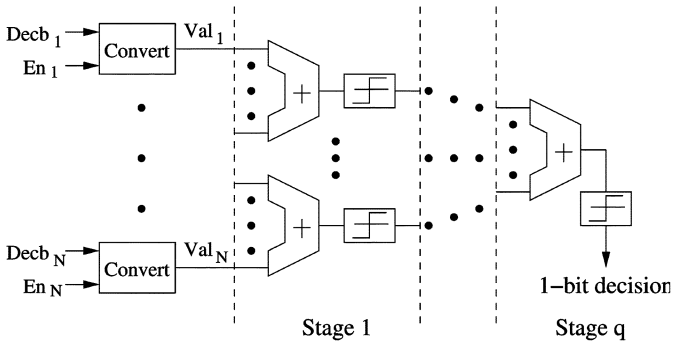


Fig. 9. Multistage one-dimensional majority vote unit block diagram.

significant global interconnection overhead and increase the latency of each modified ACS unit. By pre-computing $BM_j - w$ as normalized branch metric, it eliminates such global interconnection overhead and reduce the latency of modified ACS unit. As a result of such normalization, the metrics of the survivors are always negative.

Modified ACS (MACS) Units Array: For a trellis with N states, this decoder contains N MACS units for parallel path extension, compare & select, and path purge. Detailed description of MACS is given in Section IV-B.

Register Exchange Array (REA): It has the structure similar to that in an RE Viterbi decoder, except that it applies clock gating to disable the registers that do not store survivors.

Majority Vote Unit (MVU): This SPEC-T decoder adopts the majority vote approach to generate the decoder output. The majority vote unit architecture is described in Section IV-C.

Path Metric Search Unit (PMSU): It searches the best (minimum) path metrics among all the survivors. Given the decoder speed mismatch factor v, PMSU finishes the search once every v decoding depths. We use a partially parallel structure to implement PMSU as described in Section IV-D.

B. Modified ACS Unit

Suppose each trellis state has n incoming branches, a modified ACS unit has a structure as shown in Fig. 8. The shaded blocks are those added to the conventional ACS unit. The *compare- with-0* block, which simply observes the sign bit of the input, will make 0 pass the 2-to-1 multiplexer if its input is nonnegative. When the decoder starts, it initializes the path metric of the starting state as $-T$ (T is the pre-specified positive threshold) and the path metrics of all the other states as 0. During the decoding, the metrics of the survivors are always negative.

As illustrated in Fig. 8, it has three outputs.

i) *Path metric SM:* If the metric of the winner obtained by normal ACS operation is negative, the winner will be considered as a survivor and its metric will go through the 2-to-1 multiplexer as SM; otherwise, SM is set to 0, which represents that no survivor is led by this trellis state.

ii) *Decision symbol Dec:* Generated edtoolstvlsiby the n-input comparator, Dec is sent to the register exchange array as the decision output from present trellis state.

iii) *Enable bit En:* If the winner is a survivor, En is 1, otherwise, En is 0. It is sent to the register exchange array to turn-on/off the clock gating in the register exchange array.

Each input normalized branch metric BM_i enters a 2-to-1 multiplexer. If the associated input path metric SM_i is negative, i.e., this incoming path is a survivor, then the normalized branch metric will go through the multiplexer and be added to the path metric; otherwise, the multiplexer will output 0. This input multiplexing can reduce the switching activity: as long as the input path keeps as a nonsurvivor path (i.e., its metric is kept as 0) in successive decoding depths, there is no switching activity in the adder. Almost the same latency as a normal ACS unit can be realized since the delay of the extra multiplexing is insignificant compared with the core ACS operation.

C. Majority Vote Unit

The decoder output is determined by majority vote. Suppose the trellis has N states and the decoded symbol associated with each branch has t bits (i.e., each trellis state has 2^t incoming branches), the majority vote unit (MVU) receives N t-bit symbols from the register exchange array. Notice that each symbol may come from a survivor or a nonsurvivor, which can be determined by the associated enable bit En. Clearly, we only need to count the symbols from survivors.

The MVU contains t identical one-dimensional MVUs, each one performs the majority vote on one bit out of the t bits. Fig. 9 shows the structure of one-dimensional MVU. The input to each one-dimensional MVU are N 2-bit symbols, where each 2-bit symbol contains the enable bit En and 1 decision bit Decb. As illustrated in Fig. 9, we first convert each 2-bit symbol to an integer Val_i: if $\mathrm{En} = 0$ (i.e., the corresponding path is not a survivor), the integer Val_i is set to 0; if $\mathrm{En} = 1$ and $\mathrm{Decb} = 0$, Val_i is set to -1; if $\mathrm{En} = 1$ and $\mathrm{Decb} = 1$, Val_i is set to 1. For a complete majority vote, we need to add all the Val_is together. If the sum is positive, the output bit should be 1, otherwise the output bit is set to 0.

However, for a large trellis, such direct addition may lead to significant area/power overhead. We propose to implement each one-dimensional MVU in a multistage fashion as shown in Fig. 9. In a q-stage implementation, the number of trellis states N is factored as $N = N_1 \cdot N_2 \cdots N_q$, and the ith stage contains a group of N_i-input-1-output adder arrays. Before being sent to the next stage, the output of each adder array goes through a clipper that clips positive number to 1 and nonpositive number to -1. Compared with a complete (or 1-stage) majority vote, such multistage majority vote has the same number of adders but the wordlength of the adders are largely reduced. This may

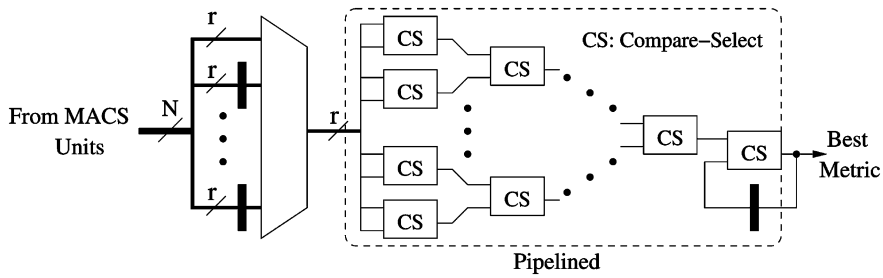


Fig. 10. Block diagram of the path metric search unit.

lead to certain power reduction at the cost of potential performance degradation. The performance degradation is negligible according to the design examples described in Section V. Finally we note that the entire MVU can be readily pipelined to support high decoding throughput.

D. Path Metric Search Unit

The path metric search unit finds the best (minimum) path metric for the speculation error correction. To reduce the area/power overhead, we realize this search operation in a time-division multiplexed fashion, as illustrated in Fig. 10. Given the N input metrics from the MACS units array, we partition them into s groups ($N \bmod s = 0$), each group contains $r = N/s$ metrics. Each clock cycle, one group of metrics is pumped into the pipelined r-input-1-output search function block that consists of compare-select (CS) elements organized in a binary-tree structure. Because most of the N input paths are nonsurvivors and have zero metrics, the switching activity in this search unit may can be insignificant.

E. A Final Remark

We note that the above parallel SPEC-T decoder may enter a catastrophic dead lock: If no survivor is generated at certain decoding depth, all the path metrics will be 0. According to the design of MACS, in all the succeeding decoding depths, all the MACS will keep the output path metric as 0 no matter how the branch metrics change because all the input branch metrics will not go through the input 2-to-1 multiplexers. Although if the threshold value of T is selected appropriately, such catastrophic dead lock can be very unusual (in fact, it never happens in the simulation of our SPEC-T convolutional code decoder design as described in Section V), we still need to completely prevent such catastrophic dead lock. To this end, we propose the following solution: First, we select a small group of trellis states $\{S_1, S_2, \ldots, S_p\}$, where $p \ll N$ and N is the total number of trellis states. Each trellis state S_i connects with S_{i+1} through a branch and S_p connects with S_1. Then we modify the MACS unit of each trellis state S_i as follows: we remove the output 2-to-1 multiplexer and the associated compare-with-0 block, and fix the output enable bit En as 1; we also remove the input 2-to-1 multiplexer that receives the path metric from state S_{i-1} (S_p if $i = 1$). In this way, we can guarantee that the catastrophic dead lock will never happen because the p selected trellis states always lead survivor paths at each decoding depth. Since the value of p is chosen to be very small, the increased power consumption due to such modification is negligible.

V. Design Examples

In this work, we use convolution code decoding as a test vehicle to demonstrate the effectiveness of the proposed parallel SPEC-T trellis decoder for the applications requiring large trellis structures. We consider the rate-1/2 convolutional codes with the constraint lengths K of 9, 8, and 7 (corresponding to the trellises with 256, 128, and 64 states, respectively). The generators are (561, 753) for $K = 9$, (247, 371) for $K = 8$, and (133, 171) for $K = 7$. For comparison, we also designed the RE Viterbi decoder counterparts, where we considered two different schemes for generating the decoder output, i.e., 1) use a multi-stage majority vote like that in the SPEC-T decoders and 2) select the oldest symbol of the survivor path led by a fixed trellis state, and the corresponding RE Viterbi decoders are denoted as MV RE Viterbi decoder and FS RE Viterbi decoder, respectively.

Design parameters of these decoders are outlined as follows: the soft input is 3-bit; the path metrics of SPEC-T decoders and RE Viterbi decoders are 6-bits and 8-bits, respectively (notice that the normalization in SPEC-T decoder helps to reduce the word-length of path metrics). The decision lengths (i.e., the length of the register exchange array) of the SPEC-T and RE Viterbi decoders are $55(K = 9)/46(K = 8)/40(K = 7)$, $55(K = 9)/46(K = 8)/40(K = 7)$ (MV RE Viterbi), and $68(K = 9)/57(K = 8)/50(K = 7)$ (FS RE Viterbi), respectively. We note that the decision lengths of MV and FS RE Viterbi decoders are selected as the minimum values that ensure the same decoding performance with small degradation from the ideal Viterbi decoding (i.e., floating point precision and infinite decision length). The decision lengths of the SPEC-T decoders are simply set equal to those of the MV RE decoders.

In all the SPEC-T and MV RE Viterbi decoders, 2-stage majority vote units are used, where the factorization of N is $256 = 8 \cdot 32$ $(K = 9)$ (i.e., the 1st stage contains 32 8-input-1-output adder arrays and the 2nd stage contains 1 32-input-1-output adder array), $128 = 8 \cdot 16$ $(K = 8)$ (i.e., the 1st stage contains 16 8-input-1-output adder arrays and the second stage contains 1 16-input-1-output adder array), and $64 = 8 \cdot 8$ $(K = 7)$ (i.e., the 1st stage contains 8 8-input-1-output adder arrays and the second stage contains 1 8-input-1-output adder array). In the SPEC-T decoders, the path metric search units partition the input data into 2 groups that share a $128(K = 9)/64(K = 8)/32(K = 7)$-input-1-output search function block. The speed mismatch factor v in the SPEC-T decoders is $7(K = 9)/7(K = 8)/6(K = 7)$. With

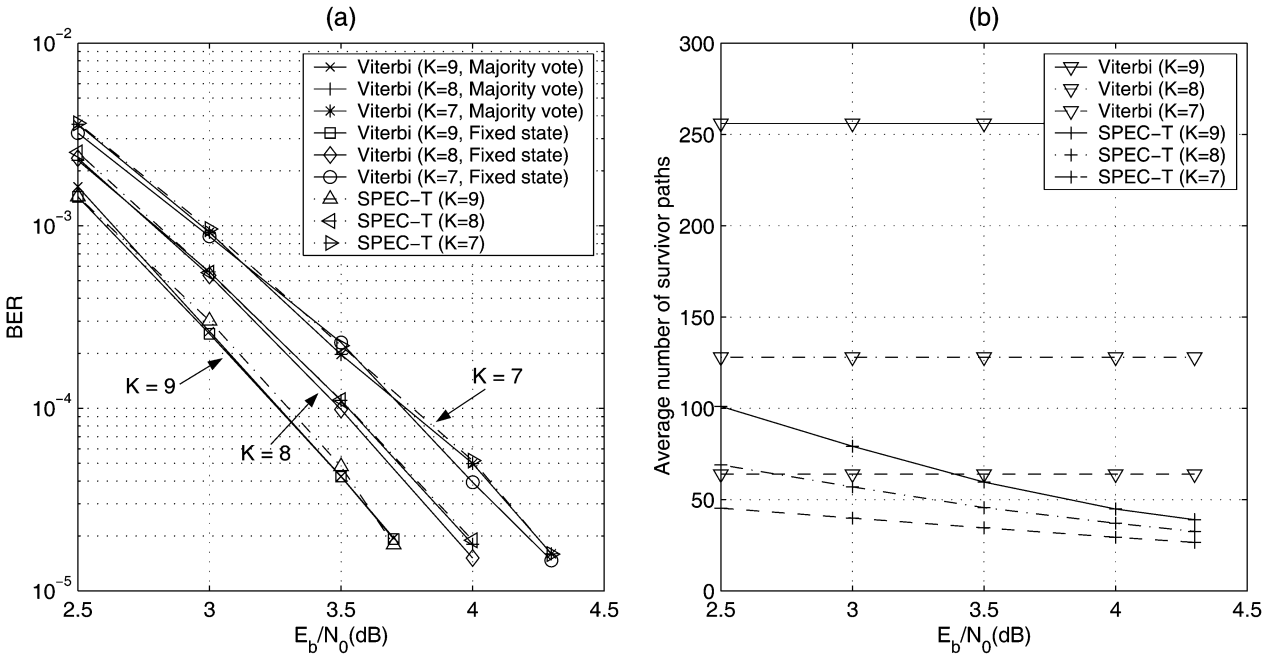


Fig. 11. Simulation results of: (a) BER versus SNR and (b) average number of survivor paths versus SNR.

TABLE I
SYNTHESIS AND POWER ESTIMATION RESULTS OF STATE-PARALLEL RE VITERBI DECODERS

	RE Viterbi (K=9)				RE Viterbi (K=8)				RE Viterbi (K=7)			
	Area (mm^2)		Power (mW)		Area (mm^2)		Power (mW)		Area (mm^2)		Power (mW)	
	MV[1]	FS[2]	MV	FS	MV	FS	MV	FS	MV	FS	MV	FS
Register exchange	1.503	1.681	219.7	243.5	0.629	0.704	87.4	102.2	0.273	0.311	37.3	45.3
ACS units	0.697	0.697	95.6	95.6	0.348	0.348	46.5	46.5	0.174	0.174	23.1	23.1
BMU	0.017	0.017	3.2	3.2	0.010	0.010	1.8	1.8	0.005	0.005	1.1	1.1
Majority vote	0.068	-	5.1	-	0.034	-	2.8	-	0.016	-	1.4	-
Total	2.285	2.395	323.6	342.3	1.021	1.062	138.5	150.5	0.468	0.491	62.9	69.5

[1]MV: Decoder output is generated by majority vote.
[2]FS: Decoder output is generated from the survivor path led by the fixed state.

the transmission power of each codeword bit normalized as 1, the threshold T is set as $27(K=9)/26(K=8)/26(K=7)$.

Assuming the signals are modulated by BPSK and transmitted over an AWGN channel, Fig. 11 shows the simulated bit error rate (BER) and average number of survivor paths of these four decoders. It clearly shows that the SPEC-T decoders can achieve almost the same decoding performance as their Viterbi counterparts with much less number of survivor paths. The SYNOPSYS tool sets are used for synthesis (Design Analyzer) and power estimation (Power Compiler) with 0.18-μm CMOS technology and 1.5-V power supply. The critical paths of all the decoders lie in the ACS or MACS units. According to the synthesis results, the RE Viterbi and SPEC-T decoders can achieve the throughput of 244 and 222 Mbps, respectively. We note that the techniques ever developed to speed up the Viterbi decoder such as lookahead (e.g., see [24], [25]) can be applied to further improve the throughput of the RE Viterbi and SPEC-T decoders in the same way. The estimated silicon area and power consumption (when decoders run at 200 Mbps) of all the decoders are listed in the Tables I–IV. Notice that the power consumption of RE Viterbi decoders almost keeps the same under different signal-to-noise ratios (SNRs), but the power consumption of SPEC-T decoders varies under different SNRs. We sample the power consumption of SPEC-T decoders at three different SNRs: 3, 3.5, and 4 dB, respectively. The tables clearly show that the developed SPEC-T decoders can effectively leverage the reduced algorithm-level computational complexity to reduce the power consumption of both register exchange array and ACS units, at the cost of larger (12% in this work) silicon area. The results clearly show that the power saving will decrease as the constraint length decreases. The main reason is that the ratio between the average numbers of survivor paths of SPEC-T and Viterbi decoders quickly increases as constraint length decreases, leading to less reduction of computational complexity and hence less (or even no) power saving. Therefore, the proposed SPEC-T decoders are more suitable to the applications demanding large trellises.

Finally, for a quick reference to the readers, Table V shows a comparison of the proposed design solution with several latest existing work on Viterbi decoder implementations. Since the ASIC design of Viterbi decoder for convolutional code decoding has been an active research area for a long time and there are

TABLE II
SYNTHESIS AND POWER ESTIMATION RESULTS OF SPEC-T DECODER ($K = 9$)

	Area (mm^2)	Power (mW)		
		@ 3dB	@ 3.5dB	@ 4dB
Register exchange	1.520	92.7	87.2	72.1
Majority vote	0.068	5.1	5.0	4.7
MACS units	0.675	59.4	57.3	50.3
Path metric search	0.226	13.3	13.2	12.7
BMU & MSU	0.021	2.6	2.5	2.5
Total[3]	2.510 (+9.85%)	173.1 (-46.51%)	165.2 (-48.95%)	142.3 (-56.03%)

[3]The number in parenthesis represents the percentage of increase or decrease against the MV RE Viterbi decoder (K=9).

TABLE III
SYNTHESIS AND POWER ESTIMATION RESULTS OF SPEC-T DECODER ($K = 8$)

	Area (mm^2)	Power (mW)		
		@ 3dB	@ 3.5dB	@ 4dB
Register exchange	0.638	52.8	51.7	44.1
Majority vote	0.034	2.8	2.8	2.7
MACS units	0.338	37.9	37.6	33.8
Path metric search	0.110	7.4	7.2	7.0
BMU & MSU	0.016	1.8	1.8	1.7
Total[4]	1.136 (+11.26%)	102.7 (-25.85%)	101.1 (-27.00%)	89.3 (-35.52%)

[4]The number in parenthesis represents the percentage of increase or decrease against the MV RE Viterbi decoder (K=8).

TABLE IV
SYNTHESIS AND POWER ESTIMATION RESULTS OF SPEC-T DECODER ($K = 7$)

	Area (mm^2)	Power (mW)		
		@ 3dB	@ 3.5dB	@ 4dB
Register exchange	0.278	34.8	31.9	29.1
Majority vote	0.016	1.4	1.4	1.4
MACS units	0.168	23.8	22.7	21.6
Path metric search	0.056	4.2	4.2	4.1
BMU & MSU	0.010	1.3	1.3	1.3
Total[5]	0.528 (+12.82%)	65.5 (+4.13%)	61.5 (-2.23%)	57.5 (-8.59%)

[5]The number in parenthesis represents the percentage of increase or decrease against the MV RE Viterbi decoder (K=7).

TABLE V
COMPARISON WITH SOME EXISTING WORK

	Proposed			[26]	[27]	[28]	[29]
Technology (μm)	0.18			0.18	0.25	0.35	0.5
# of states	256	128	64	64	256	64	256
Throughput (Mbps)	200	200	200	54	20	100	2
Power[6](mW)	165.2	101.1	61.5	68	450	120	10

[6]In the context of proposed design, the power values are those estimated under 3.5dB SNR.

a large amount of existing work, this by no means suggests a comprehensive comparison.

VI. CONCLUSIONS

This paper presents techniques at the algorithm and VLSI architecture levels to realize parallel limited search decoder VLSI implementation. Based on the well-known T-algorithm, we developed a SPEC-T algorithm that inherently provides great potential of parallel high-throughput implementation. At the architecture level, we develop a parallel SPEC-T decoder architecture based on the conventional state-parallel register exchange Viterbi decoder architecture. Using the convolutional code decoding as a test vehicle, we demonstrated significant power savings of this proposed SPEC-T decoder compared with the Viterbi decoder. This work provides an unique opportunity to exploit the attributes of the T-algorithm to reduce the trellis decoder power consumption while achieving almost the same throughput and decoding performance with a state-parallel Viterbi decoder. It is our hope that this work will inspire the rethinking of the potential of the limited search trellis decoding in the real-world applications and motivate more future research work in this area.

REFERENCES

[1] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 218–278, Mar. 1973.

[2] J. B. Anderson and E. Offer, "Reduced-state sequence detection with convolutional codes," *IEEE Trans. Inf. Theory*, vol. 40, pp. 965–972, May 1994.

[3] J. B. Anderson, "Limited search trellis decoding of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 35, pp. 944–955, Sep. 1989.

[4] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory*, vol. IT-9, pp. 64–74, Apr. 1963.

[5] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, pp. 675–685, Nov. 1969.

[6] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. 32, pp. 169–176, Feb. 1984.

[7] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. 38, pp. 3–12, Jan. 1990.

[8] S. K. Singh, P. Thiennviboon, R. O. Ozdag, S. Tugsinavisut, P. A. Beerel, and K. M. Chugg, "Algorithm and circuit co-design for a low-power sequential decoder," in *Proc. 33rd Asilomar Conf. on Signals, Systems, and Computers*, Oct. 1999, pp. 389–394.

[9] C. Y. Lee, F. Catthoor, and H. De Man, "Breaking the bottleneck of sequential decoding for high-speed digital communication," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, Apr. 1991, pp. 1213–1216.

[10] T. M. Gould and J. H. Harris, "Single-chip design of bit-error-correcting stack decoders," *IEEE J. Solid-State Circuits*, vol. 27, pp. 768–775, May 1992.

[11] P. Lavoie, D. Haccoun, and Y. Savaria, "A systolic architecture for fast stack sequential decoders," *IEEE Trans. Commun.*, vol. 42, pp. 324–335, Feb.–Apr. 1994.

[12] W.-W. Yang, L.-F. Jeng, and C.-Y. Lee, "Design of a fast sequential decoding algorithm based on dynamic searching strategy," in *Proc. IEEE Int. Symp. on Circuits and Systems*, May 1994, pp. 165–168.

[13] C.-Y. Lee, "A cost-effective VLSI architecture for high-throughput sequential decoder," in *Proc. IEEE Int. Symp. on Circuits and Systems*, May 1996, pp. 328–331.

[14] S. J. Simmons and S. Tsui, "A reduced-power algorithm and VLSI architecture for sequential decoding," in *Proc. Canadian Conf. on Electrical and Computer Engineering*, Mar. 2000, pp. 48–52.

[15] S. J. Simmons, "A nonsorting VLSI structure for implementing the (M, L) algorithm," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 538–546, Apr. 1988.

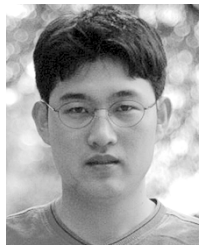
[16] ——, "A bitonic-sorter based VLSI implementation of the M-algorithm," in *Proc. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, Jun. 1989, pp. 337–340.

[17] P. A. Bengough and S. J. Simmons, "Sorting-based VLSI architectures for the M-algorithm and T-algorithm trellis decoders," *IEEE Trans. Commun.*, vol. 43, pp. 514–522, Feb. 1995.

[18] M.-H. Chan, W.-T. Lee, M.-C. Lin, and L.-G. Chen, "IC design of an adaptive Viterbi decoder," *IEEE Trans. Consum. Electron.*, vol. 42, pp. 52–62, Feb. 1996.

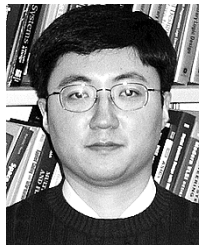
[19] R. Henning and C. Chakrabarti, "An approach for adaptively approximating the Viterbi algorithm to reduce power consumption while decoding convolutional codes," *IEEE Trans. Signal Processing*, vol. 52, no. 5, pp. 1443–1451, May 2004.

[20] C. Rader, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. 29, no. 9, pp. 1399–1401, Sep. 1981.
[21] P. J. Black and T. H. Meng, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Apr. 1993, pp. 433–436.
[22] E. Boutillon and N. Demassieux, "High speed low power architecture for memory management in a Viterbi decoder," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 4, May 1996, pp. 284–287.
[23] H.-L. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Processing Mag.*, vol. 12, pp. 42–52, Sep. 1995.
[24] G. Fettweis and H. Meyr, "High-speed parallel Viterbi decoding: Algorithm and VLSI-architecture," *IEEE Commun. Mag.*, vol. 29, pp. 46–55, May 1991.
[25] K. K. Parhi, "High-speed VLSI architectures for Huffman and Viterbi decoders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, pp. 385–391, Jun. 1992.
[26] C.-C. Lin *et al.*, "Design of a power-reduction Viterbi decoder for WLAN applications," IEEE Trans. Circuits Syst. I, Reg. Papers, 2005, to be published.
[27] X. Liu and M. C. Papaefthymiou, "Design of a 20-mb/s 256-state Viterbi decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 965–975, Dec. 2003.
[28] T. Gemmeke, M. Gansen, and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 37, no. 7, pp. 941–948, Jul. 2002.
[29] 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.



Fei Sun received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2000 and 2003, respectively. He is working toward the Ph.D. degree in Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY.

His current research interests include design of VLSI architectures and circuits for wireless communication systems and on-chip error-correction system design for semiconductor memory.



Tong Zhang (S'98–M'02) received the B.S. and M.S. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering at the University of Minnesota in 2002.

Currently he is an assistant professor in Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY. His current research interests include design of VLSI architectures and circuits for digital signal processing and communication systems, with the emphasis on error-correcting coding, multiple input multiple output (MIMO) signal processing, and asynchronous VLSI signal processing.