Low-Power State-Parallel Relaxed Adaptive Viterbi Decoder

Fei Sun, *Student Member, IEEE*, and Tong Zhang, *Member, IEEE*

***Abstract*—Although it possesses reduced computational complexity and great power saving potential, conventional adaptive Viterbi algorithm implementations contain a global best survivor path metric search operation that prevents it from being directly implemented in a high-throughput state-parallel decoder. This limitation also incurs power and silicon area overhead. This paper presents a modified adaptive Viterbi algorithm, referred to as the relaxed adaptive Viterbi algorithm, that completely eliminates the global best survivor path metric search operation. A state-parallel decoder VLSI architecture has been developed to implement the relaxed adaptive Viterbi algorithm. Using convolutional code decoding as a test vehicle, we demonstrate that state-parallel relaxed adaptive Viterbi decoders, versus Viterbi counterparts, can achieve significant power savings and modest silicon area reduction, while maintaining almost the same decoding performance and very high throughput.**

***Index Terms*—Adaptive Viterbi algorithm, low power, T-algorithm, very large-scale integration (VLSI) architecture.**

I. Introduction

THE adaptive Viterbi algorithm [1], which combines the Viterbi algorithm with the principle of T-algorithm [2], has a computational complexity which adapts to run-time channel conditions. This approach has a great potential of realizing significant power savings [3]. In contrast to the Viterbi algorithm, in the adaptive Viterbi algorithm, the winner path at each trellis state does not necessarily become a survivor path, i.e., only those whose path metrics are better than a global *nonsurvivor purge limit* will be fed to the next decoding depth as survivors. The nonsurvivor purge limit is determined by the overall best winner path at each decoding depth and varies from one decoding depth to the next. Due to its serial nature, the *search-for-the-best-winner* operation tends to have a much longer latency than the other operations within the recursive decoding loop. For a VLSI implementation of the adaptive Viterbi algorithm, the long latency due to such a search operation may be concealed by using state-serial decoder architectures [4], [5] or partially state-parallel decoder architecture [6] at the cost of achievable decoding throughput. The authors of [7], [8] developed a configurable state-parallel adaptive Viterbi decoder that can effectively support a large trellis structure but is subject to a relatively long recursive decoding datapath delay. As a result, the achievable throughput demonstrated for adaptive Viterbi decoders in the open literature [i.e., up to few megabits per second (Mbps)] is far less than that of a state-parallel Viterbi decoder[1] that can readily achieve a throughput of several hundred Mbps. This leaves an open problem of leveraging the power-saving potential of the adaptive Viterbi algorithm in applications demanding very high decoding throughput, e.g., several hundred Mbps.

As an attempt to tackle this challenge, the authors [9] recently proposed a modified adaptive Viterbi algorithm and a state-parallel decoder architecture. The key feature of the architecture is the movement of the search-for-the-best-winner operation to the outside of the main recursive decoding loop. Therefore, a state-parallel decoder may achieve a throughput comparable to that of its state-parallel Viterbi counterpart. However, the search-for-the-best-winner operation still incurs noticeable power and silicon area overhead. This will inevitably degrade the power saving efficiency, particularly for small and moderate trellises such as 64-state and 128-state. For example, as shown in [9], for the decoding of a 64-state convolutional code, the state-parallel modified adaptive Viterbi decoder may even consume more power than its Viterbi counterpart at relatively low signal-to-noise ratio (SNR). Moreover, the synthesis results presented in [9] show that the decoders may occupy $>10\%$ more silicon area than their Viterbi counterparts.

To solve the above drawbacks of the design solution presented in [9], in this work, we developed a new approach to modify the adaptive Viterbi algorithm. We refer the developed algorithm as the *relaxed* adaptive Viterbi algorithm. As the key difference from the design solution in [9], the relaxed adaptive Viterbi algorithm completely eliminates the search-for-the-best-winner operation, and hence can realize much better power savings and silicon area reduction. We further developed the VLSI architecture of a state-parallel relaxed adaptive Viterbi decoder. Compared with its state-parallel Viterbi counterpart, a state-parallel relaxed adaptive Viterbi decoder can realize significant power savings and modest silicon area reduction, while maintaining almost the same decoding performance and throughput. Using 0.13-μm CMOS standard cell and static random access memory (SRAM) libraries and Synopsys tools for synthesis, layout, and post-layout power estimation, we designed relaxed adaptive Viterbi decoders for rate-1/2 convolutional codes with 64, 128, and 256 states, respectively. Compared to their Viterbi counterparts, the relaxed adaptive Viterbi decoders realize up to 7% silicon area reduction, 73% of power savings on the decoding computation,[2] and 81% and 48% of overall decoder power savings if

Manuscript received November 28, 2005; revised May 1, 2006, and September 1, 2006. This work was supported by the National Science Foundation under Grant ECS-0522457. This paper was recommended by Associate Editor Z. Wang.

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/TCSI.2007.890617

[1]For an N-state trellis, a state-parallel Viterbi decoder implements N add-compare-select (ACS) units that operate in parallel.

[2]The total power consumption of a convolutional code decoder contains two parts, including (1) power consumed by *decoding computation* including ACS computation, branch metric computation, etc., and (2) power consumed by *decoder output generation* that is carried out by a survivor memory unit.

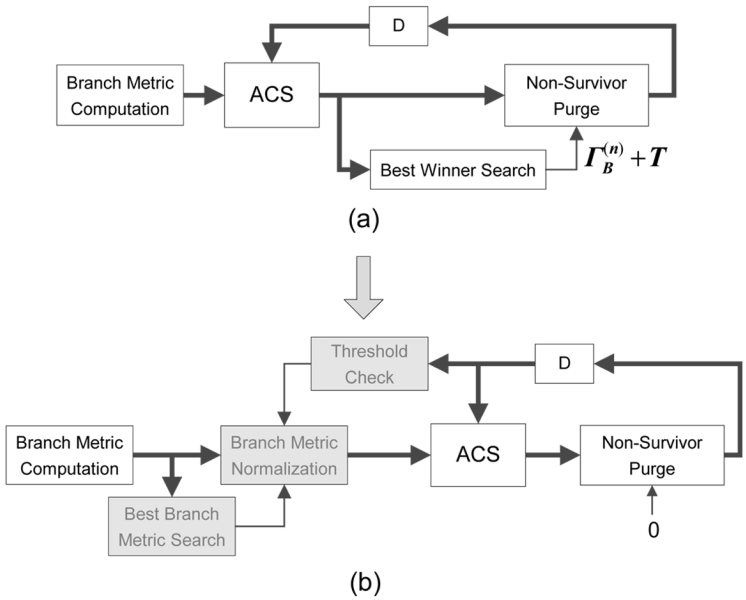

Fig. 1. Recursive data flow diagrams of (a) original adaptive Viterbi algorithm and (b) proposed relaxed adaptive Viterbi algorithm.

the register-exchange and TB approaches are used to implement the survivor memory unit, respectively. These rate-1/2 code decoders operate at 200 MHz and achieve 200 Mbps decoding throughput.

The remainder of this paper is organized as follows. Sections II and III present the proposed relaxed adaptive Viterbi algorithm and the corresponding VLSI architecture. Design examples of state-parallel relaxed adaptive Viterbi convolutional code decoders are presented in Section IV, and the conclusions are drawn in Section V.

II. Relaxed Adaptive Viterbi Algorithm

A. Algorithm Formulation

Fig. 1(a) shows the recursive data flow diagram of an adaptive Viterbi algorithm, which adds two functional blocks, including the best winner search and nonsurvivor purge [2], into the original Viterbi algorithm. At the lth decoding depth, after all the ACS units determine their own local winners, the best winner search block finds the one having the best (minimum) path metric among all the winners, denoted as $\Gamma_B^{(l)}$, and the nonsurvivor purge block deletes the local winners whose metric $\Gamma^{(l)} \geq \Gamma_B^{(l)} + T$ and feeds the others as survivors to the next decoding depth, where T is a fixed positive number. The value of $\Gamma_B^{(l)} + T$ is the nonsurvivor purge limit. Notice that the essential goal of the search-for-the-best-winner operation in the adaptive Viterbi algorithm is to determine the nonsurvivor purge limit at each decoding depth, which may change from one depth to the next. The value of T determines the width of the survivor path retention window (i.e., the region between the nonsurvivor purge limit and the best winner path metric). Due to the serial nature of the search operation, the best winner search block inevitably incurs a large delay in the recursive decoding datapath, which makes the adaptive Viterbi algorithm not suitable for a state-parallel decoder structure.

In this work, we developed a method to eliminate the search-for-the-best-winner operation and hence enable the high-throughput state-parallel adaptive Viterbi decoder implementation. The basic idea is to *dynamically normalize* the branch metrics in such a way that the metric of the overall best winner, i.e., $\Gamma_B^{(l)}$, is almost always very close to $-T$, which means the nonsurvivor purge limit, i.e., $\Gamma_B^{(l)} + T$, is almost always very close to zero. As a relaxation, we simply fix the nonsurvivor purge limit as zero at each decoding depth. This directly eliminates the search-for-the-best-winner operation in the recursive decoding datapath, and the resulted algorithm is referred to as the relaxed adaptive Viterbi algorithm. The branch metric dynamic normalization is realized by the three shaded functional blocks as shown in Fig. 1(b), which are described as follows.

- *Threshold Check*: It checks whether at least one survivor has a metric less than $-T+r$, where r is a positive number that is much less than T. If yes, it outputs a zero, otherwise, it outputs r.
- *Best Branch Metric Search*: At the lth decoding depth, it simply finds the best (minimum) branch metric, denoted as $BM_B^{(l)}$, among the all the present branch metrics.
- *Branch Metric Normalization*: At each depth, given the input $d^{(l)}$, which is either zero or r, from the Threshold Check and the input $BM_B^{(l)}$ from the Best Branch Metric Search, it subtracts $BM_B^{(l)} + d^{(l)}$ from all the branch metrics and feeds these normalized branch metrics to the succeeding ACS operations.

If at least one survivor has a metric less than $-T+r$ (i.e., the metric of the best survivor is very close to $-T$ since r is much less than T), the normalized branch metrics will be nonnegative with the minimum value of zero, and the path metrics will monotonically increase. If none of the survivors has a metric less than $-T+r$ (i.e., the metric of the best survivor is not very close to $-T$), we bias the branch metric normalization by r to

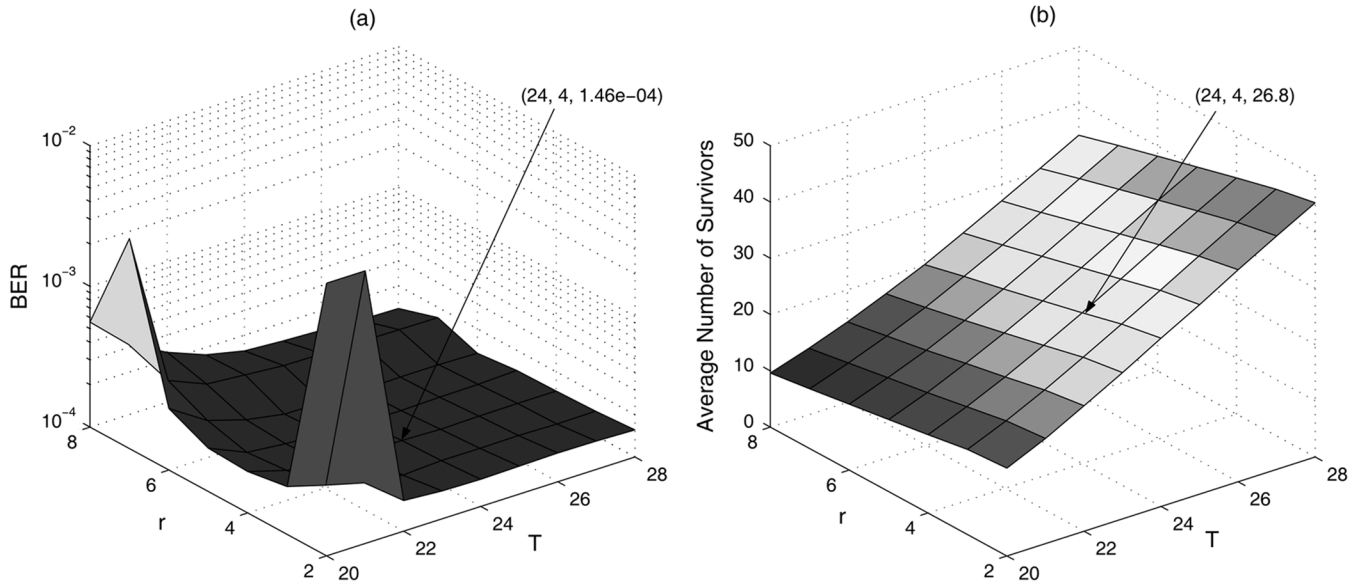

Fig. 2. Simulated BER and average number of survivors as a function of T and r.

push the path metrics toward $-T$. With appropriate selection of r, we can dynamically adjust the best metric almost always very close to $-T$.

B. Decoding Performance

All the existing variants of adaptive Viterbi algorithms (including the one presented above) may lose the correct path [i.e., the maximum likelihood (ML) path] during the decoding due to their nature of nonexhaustive trellis search. This inevitably results in decoding performance degradation compared with Viterbi algorithm. As discussed in [1], in order to quickly recover the correct path once it has been lost, the width of the survivor path retention window (i.e., the value of T in conventional adaptive Viterbi algorithms) has to be large enough and the maximum allowable number of survivors, denoted as $N_{\max}$, should be equal to the total number of trellis states (i.e., 2^{K-1}, where K is the constraint length of the convolutional code). Thus, the conventional adaptive Viterbi decoder reduces power by choosing an appropriate T that will meet the performance constraints of the system while allowing up to 2^{K-1} survivors to be kept at each decoding depth [1], [3].

In the above presented relaxed adaptive Viterbi algorithm, the width of the survivor path retention window is no longer a fixed value but is almost always very close to T due to the dynamic branch metric normalization with the parameter r. Furthermore, similar to [1], $N_{\max}$ in the relaxed adaptive Viterbi algorithm equals to the total number of trellis states. Thus, the performance of the relaxed adaptive Viterbi algorithm solely depends on T and r. Similar to the conventional adaptive Viterbi algorithms [1], [7], [3], we apply computer simulations to evaluate the effect of T and r on the soft-decision decoding performance as illustrated in the following example.

Example 2.1: Consider a rate-1/2 convolutional code with constraint length $K = 7$ (i.e., the corresponding trellis has 64 states). Assuming the convolutional code is modulated by binary phase shift keying (BPSK) and transmitted over an additive white Gaussian noise (AWGN) channel, Fig. 2 shows the fixed-point simulations on the bit-error rate (BER) and average number of survivors at the SNR of 3.5 dB for various values of T and r (where the finite word-length of soft input and path metrics are 3 and 6 bits, respectively). Under the same SNR, the ideal Viterbi decoding (i.e., floating point precision and infinite decision length) has a BER of $9.15\mathrm{e}-5$. Fig. 2 shows that the decoding performance with $T \geq 23$ and $2 \leq r \leq 6$ is very close to that of ideal Viterbi decoder. Considering the average number of survivor paths that determines the power saving potential, one may choose $T = 24$ and $r = 4$ as the final parameters in this case.

TABLE I
EMPIRICALLY SELECTED VALUES OF T AND r

	Rate-1/4		Rate-1/2	Rate-2/3	
	K=7, 8	K=9	K=7, 8, 9	K=7, 8	K=9
T	44	48	24	20	22
r	8	8	4	4	6

To further demonstrate the decoding performance and average number of survivors of the proposed relaxed adaptive Viterbi algorithm, we carried out computer simulations on various convolutional codes described as follows: We considered three different code rates, including 1/4, 1/2, and 2/3, and three different constraint lengths K, including 7, 8, and 9 (the corresponding trellises have 64, 128, and 256 states, respectively). In the simulation, the relaxed adaptive Viterbi algorithm generates the output as follows: it first traces back L depths to determine a merged state from which it then traces back D depths to generate D output symbols. The parameters of $\{L, D\}$ are $\{48, 24\}$ for $K = 7$, $\{56, 28\}$ for $K = 8$, and $\{64, 32\}$ for $K = 9$, respectively. In the fixed-point simulation, we use 3-bit soft input and 6-bit path metric, and the parameters of $\{T, r\}$ are listed Table I.

Assuming these convolutional codes are modulated by BPSK and transmitted over an AWGN channel (Fig. 3) shows the simulated BER and average number of survivors of the relaxed adap-

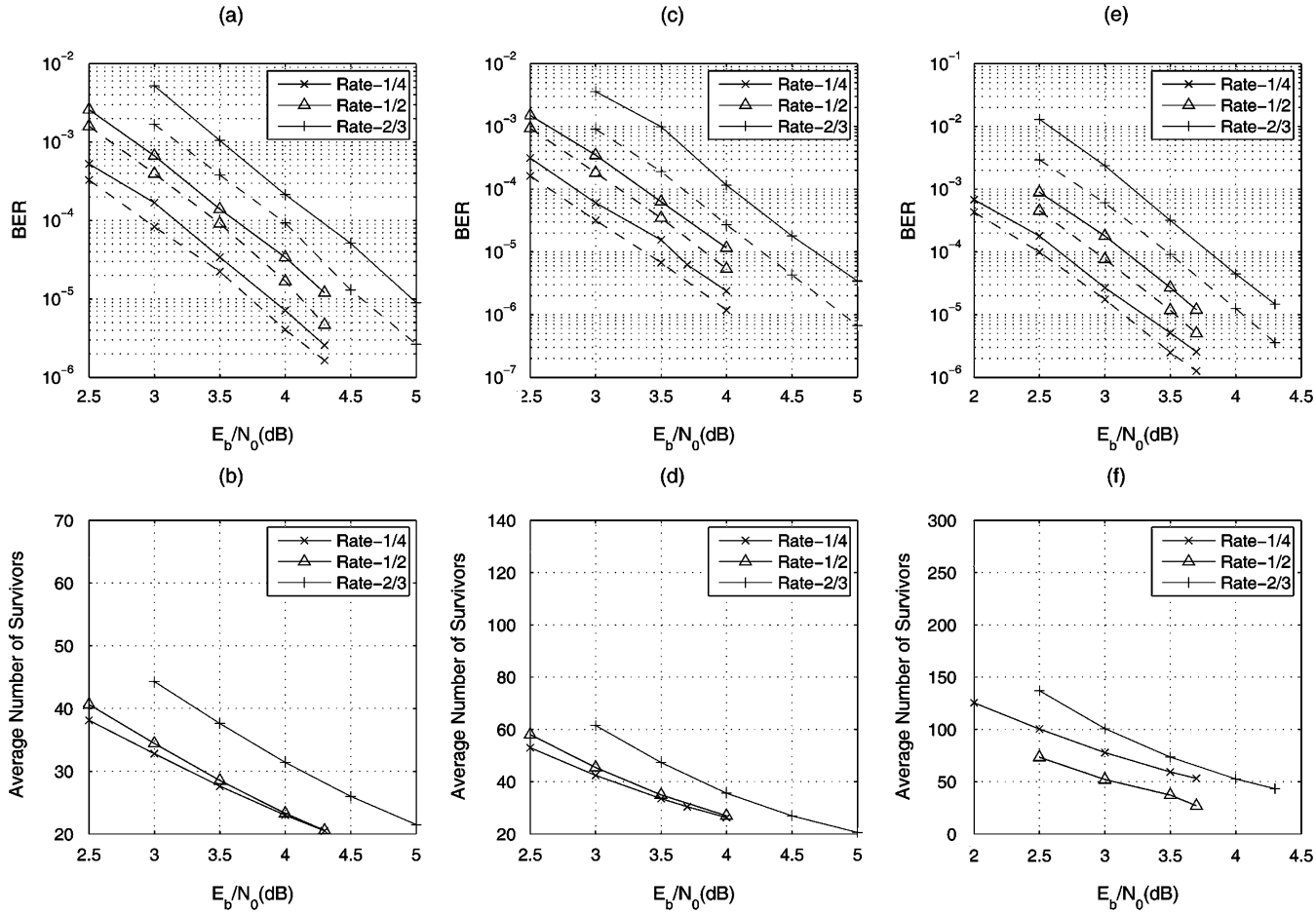

Fig. 3. Simulated BER and average number of survivors for various code rates and constraint lengths. The solid and dashed curves in (a), (c), and (e) correspond to the relaxed adaptive Viterbi and ideal Viterbi decoders, respectively. (a) $K = 7$. (b) $K = 7$. (c) $K = 8$. (d) $K = 8$. (e) $K = 9$. (a) $K = 9$.

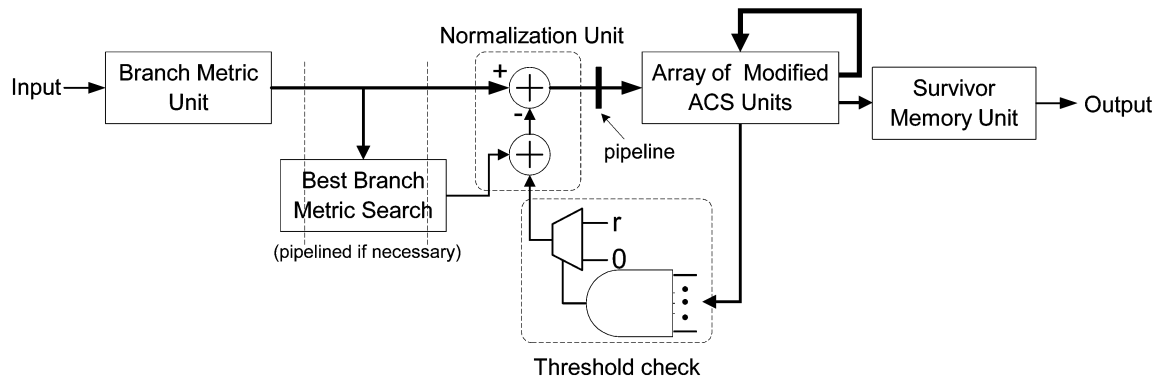

Fig. 4. Structure of a state-parallel relaxed adaptive Viterbi decoder.

tive Viterbi decoders. In each case, we also show the performance of ideal Viterbi algorithm, where *ideal* means the use of floating point precision and infinite decision length.

III. State-Parallel Relaxed Adaptive Viterbi Decoder Design

The relaxed adaptive Viterbi algorithm can be directly mapped onto a state-parallel decoder with the structure as shown in Fig. 4, which is very similar to that of a state-parallel Viterbi decoder. Similar to a conventional Viterbi decoder, the branch metric unit 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, This simplification will not (largely) affect the decoding performance [10].

The best branch metric search block simply finds the best (minimum) branch metric among the all the present branch metrics. Since it is located outside the recursive datapath, it can be directly pipelined if necessary. The *Threshold Check* function

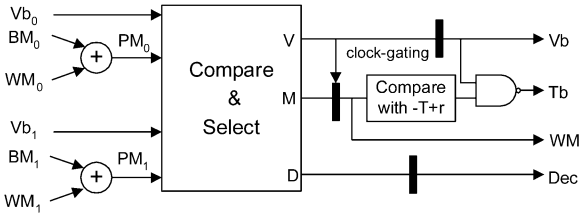

Fig. 5. Architecture of a modified ACS unit.

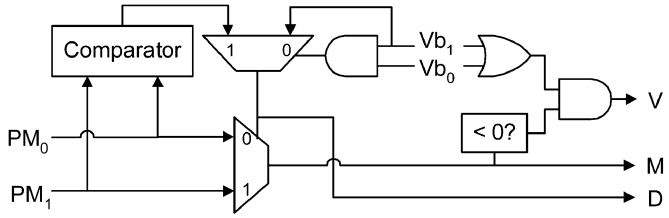

Fig. 6. Example *Compare and Select* block (note that the compare-with-0 block generates a 1 if M is negative).

in the relaxed Viterbi algorithm is implemented as follows: the compare-with-$(-T+r)$ is realized by each individual modified ACS unit, and the pass/fail decisions from all the modified ACS units AND together to determine whether a zero or r should be sent to the normalization unit, as illustrated in Fig. 4. In the following, we will elaborate the architectures of the other two functional blocks: modified ACS unit and survivor memory unit.

A. Modified ACS

For a trellis with N states, a state-parallel relaxed adaptive Viterbi decoder contains N modified ACS units that operate in parallel. When the decoder starts, it initializes the path metric of the starting state as $-T$ and the path metrics of all the other states as 0. As discussed in Section II, we fix the nonsurvivor purge limit to zero so that only winners with negative path metrics can become survivors. Suppose each trellis state has two incoming/outgoing branches, the modified ACS unit has a structure as shown in Fig. 5.

It receives two pairs of $\{\mathrm{BM}_i, \mathrm{WM}_i, Vb_i\}, i \in \{0,1\}$, where each BM_i is a normalized branch metric sent from the branch metric normalization unit, each WM_i is a winner path metric fed back from one modified ACS unit, and each 1-bit Vb_i indicates whether this incoming winner is a survivor or not. Each modified ACS unit has four outputs, including: 1) winner path metric WM; 2) validity bit Vb ($Vb = 1$ means that a survivor is generated from the present trellis state); 3) decision bits Dec to indicate which incoming path is the winner and 4) threshold check result Tb ($Tb = 0$ means that the corresponding winner path is a survivor and its metric is less than $-T + r$).

Since only one incoming path becomes the winner, the *Compare and Select* block selects the best one from the path metrics (i.e., PM_i as shown in Fig. 5) obtained from survivors. Fig. 6 shows the architecture of a *Compare and Select* block, which can be explained as follows.

- If both incoming paths are survivors (i.e., $Vb_0 = Vb_1 = 1$), then, similar to the conventional Viterbi decoder, the output of the comparator will select the best path metric as the output winner metric M. The output V will be 1 if the winner metric is negative, otherwise it will be 0.
- If only one incoming path is a survivor (i.e., Vb_0 XOR (Exclusive-OR) $Vb_1 = 1$), then the output of the comparator will be ignored and the path metric obtained from the survivor will go through the multiplexer and become the output winner metric M. Again, the output V will be 1 if the winner metric is negative, otherwise it will be 0.
- If neither incoming path is a survivor (i.e., $Vb_0 = Vb_1 = 0$), then the output V will be 0. In this scenario, the winner path metric can be set to be either PM_0, PM_1, or any other arbitrary value (in our design, according the architecture in Fig. 6, the winner path metric will be equal to PM_0), which will not affect the succeeding decoding.

As shown in Fig. 5, we apply clock-gating on the winner path metric output WM, i.e., if the current winner path is not a survivor (i.e., $V = 0$), then we keep the WM output unchanged in order to reduce the switching activity in the subsequent decoding recursion.

B. Survivor Memory Unit

As extensively discussed in the literature (e.g., [11]–[14]), the survivor memory unit can be designed in two different styles, register exchange (RE) and trace-back (TB), targeting different trade-offs among power, silicon area, and throughput. In general, RE can easily support very high decoding throughput but occupies larger silicon area and tends to consume more power, while TB requires less silicon area and power but may not readily support very high decoding throughput. In the following, we will discuss how to design the survivor memory unit of a state-parallel relaxed adaptive Viterbi decoder based on either an RE or TB design style.

RE-Based Design: Assuming each trellis state only has two incoming/outgoing branches, Fig. 7 shows the structure of an RE-based design solution by introducing clock-gating into the RE array followed by majority vote unit. The validity bits from the modified ACS units array directly control the clock-gating for power reduction. Since the average number of survivors can be much less than the total number of trellis states in adaptive Viterbi decoding, this may lead to significant power savings. The decoder output is determined by a majority vote unit that only counts decision symbols from survivors. In this work, we use the multistage majority vote unit design approach proposed in [9]. The length of the register exchange array is called decision length L.

TB-Based Design: To support high throughput in a state-parallel decoder, the TB-based survivor memory unit must provide large enough memory access bandwidth. One natural approach to increase the access bandwidth is to use a bank of memories that can be accessed concurrently, e.g., the k-pointer even and odd schemes presented in [13]. As the cost of access bandwidth increases, the required SRAM resource increases accordingly. These techniques can be equally applied to the relaxed adaptive Viterbi decoder with only one subtle difference: In the conventional Viterbi decoder, we can start the TB from any arbitrary trellis state; while for the relaxed adaptive Viterbi decoder, since not all the trellis states lead to survivors, we need to ensure that TB always starts from a state leading to a survivor in the present decoding depth. Suppose the trellis has N states, we may use an N-to-$\lceil \log_2 N \rceil$ priority encoder with the input of

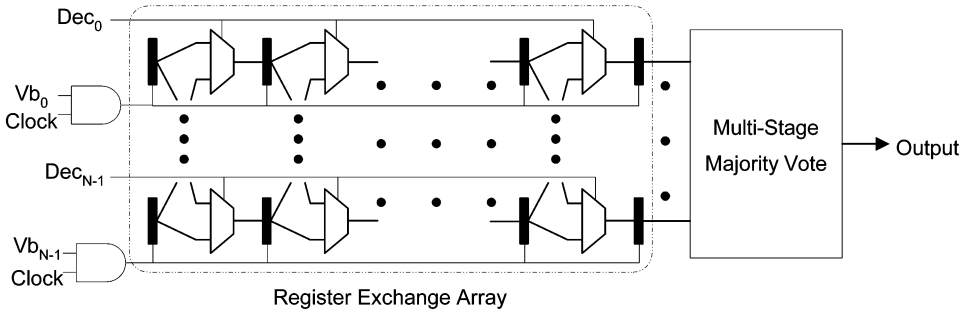

Fig. 7. Structure of RE-based design.

N validity bits. The output of the encoder will always point to a state leading to a survivor, from which we may start the TB. For a TB-based design to generate the decoder output, it first traces back decision length L depths to determine a merged state from which it then traces back D steps to generate D output symbols. To realize a high throughput, the value of D is typically not small, e.g., it may equal to L or $L/2$ [15], [16]. In this work, for the ASIC design of TB-based decoders as described in Section IV, we used a 3-pointer even scheme to design the TB-based survivor memory unit and set $D = L/2$.

C. Discussion

For the above proposed state-parallel relaxed adaptive Viterbi decoder, we argue that, compared with its state-parallel Viterbi decoder counterpart, it can achieve:

Significant Power Saving: The power saving is gained from both decoding computation and decoder output generation.

1) The power saving on decoding computation is realized by clock-gating the metric output of nonsurvivors, as shown in Fig. 5, which leads to largely reduced switching activity and hence power consumption in subsequent ACS computations due to the significantly reduced number of survivors.
2) The power saving on decoder output generation depends on the survivor memory unit implementation style: a) If we use the RE style, we can simply clock-gate the registers associated with nonsurvivors to reduce the switching activity and hence the power consumption and b) If we use the TB style, the power saving will largely depend on how the SRAM is implemented. We note that bit-line charging/discharging in memory write operations tends to dominate the power consumption of SRAM [17]. Therefore, if the memory can be specially customized so that, in the write operation, one can determine which bit(s) in one word are written into memory cells and force the bit-lines associated with the other bits into read mode, a certain amount of power can be saved by only writing the decision bits from survivors. If such a specially customized memory is not available, however, no power saving can be realized in the decoder output generation.

Modest silicon area reduction: Contrary to the first impression that the relaxed adaptive Viterbi decoder must occupy larger area due to the extra functional blocks, relaxed adaptive Viterbi decoders may have modestly reduced silicon area because the branch metric normalization can help to reduce the finite word-length of the path metrics.

TABLE II
DESIGN PARAMETERS

	Viterbi	Relaxed Adaptive Viterbi
L in RE Decoders	40 (K=7), 46 (K=8), 55 (K=9)	
$\{L, D\}$ in TB Decoders	$\{48, 24\}$ (K=7), $\{56, 28\}$ (K=8), $\{64, 32\}$ (K=9)	
Soft Input	3 bits	
Path metric[3]	8 bits	6 bits
Others	N/A	$T = 24$, $r = 4$

[3]In the Viterbi decoder, the computation of path metrics is based on modulo arithmetic, where the configuration of 3-bit soft input and 8-bit path metric is widely used in open literature, e.g., see [16].

Moreover, as demonstrated in Section IV, the state-parallel relaxed adaptive Viterbi decoders can achieve almost the same decoding performance (with appropriate selection of T and r) and decoding throughput, compared with their state-parallel Viterbi counterparts. The reason for the latter can be briefly explained as follows:

1) The reduced finite word-length of path metrics can compensate for the latency overhead incurred by the slightly more complex operation in the ACS computation;
2) The threshold check operation will not incur a speed bottleneck because the involved operation is very simple and it can be directly pipelined if necessary (in the design examples described later, the overall decoder critical path always lies in ACS computation).

Finally, we note that, due to their similar architectures, the circuit-level design/optimization techniques developed for state-parallel Viterbi decoders could also be applied to state-parallel relaxed adaptive Viterbi decoders.

IV. Design Examples

For the purpose of demonstration, we designed state-parallel relaxed adaptive Viterbi decoders for rate-1/2 convolutional codes with constraint lengths K of 7, 8, and 9 (corresponding to the trellises with 64, 128, and 256 states, respectively). The code generators are (133, 171) for $K = 7$, (247, 371) for $K = 8$, and (561, 753) for $K = 9$. We considered both RE-based and TB-based design approaches for the survivor memory unit, where the TB-based approach uses the 3-pointer even scheme presented in [13].

For comparison, we also designed state-parallel Viterbi counterparts. We note that an RE Viterbi decoder can generate output in two possible approaches: 1) output the last (oldest) symbol of the survivor path led by a fixed trellis state or 2) apply a majority

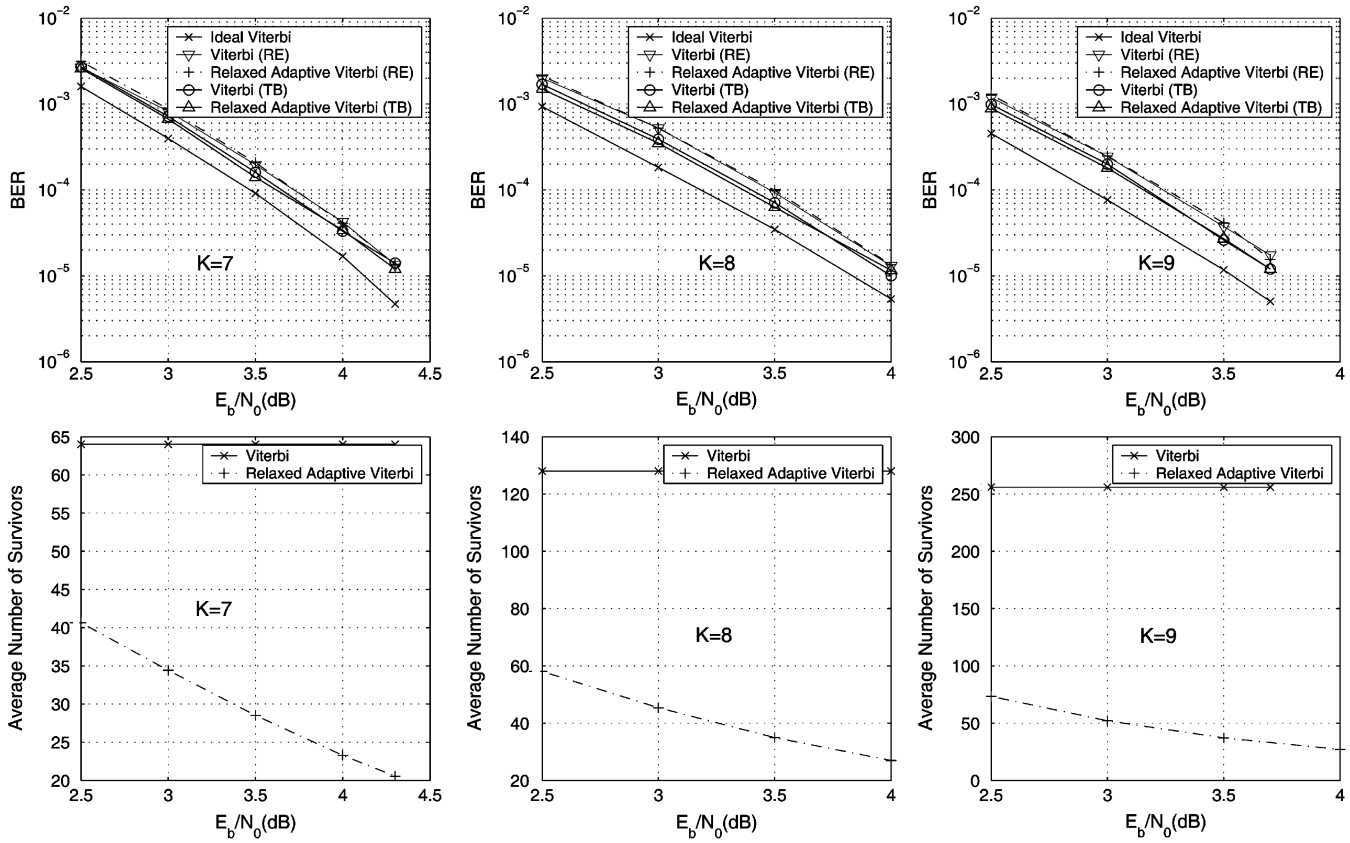

Fig. 8. Simulated BER and average number of survivors for rate-1/2 convolutional codes with various constraint lengths.

vote on the last symbols of all the survivors. To realize the same decoding performance, the latter approach requires a shorter decision length at the cost of implementing a majority vote. As discussed in [18], the majority vote can be slightly more power-efficient. Therefore, the RE Viterbi decoders use the same multistage majority vote as in the relaxed adaptive Viterbi decoders. The decoder design parameters are outlined in Table II.

The parameter L, called decision length, presents the length of the register-exchange array in RE decoders and the number of TB depths for searching the merged trellis state in TB decoders. In TB decoders, another parameter D represents the number of symbols output by each TB. Moreover, we note that the branch metric normalization in relaxed adaptive Viterbi decoders helps to reduce the finite word-length of the path metrics.

Assuming these convolutional codes are modulated by BPSK and transmitted over an AWGN channel, Fig. 8 shows the fixed-point simulations of BER and average number of survivors of the relaxed adaptive Viterbi decoders. In each case, we also show the performance of ideal Viterbi decoding (i.e., floating point precision and infinite decision length). The relaxed adaptive Viterbi decoders can achieve almost the same decoding performance as their Viterbi counterparts, while incurring a lower number of survivors.

For the ASIC design of these decoders, we use 0.13-μm CMOS standard cell and SRAM libraries, and Synopsys tools are used for synthesis (Design Compiler), layout (Astro), and post-layout power estimation (Prime Power). We used the worst-case libraries during synthesis and layout. We set the target throughput as 200 Mbps (with the power supply of 1.08 V) and the results show that all the decoders can meet this target with similar timing slacks. It should be noted that the highest achievable throughput of the decoders with different constraint lengths may be different in practice due to the potentially different contribution of routing delay after customized place and route optimization, especially in sub-100 nm technology where interconnect delay is becoming more significant. In this work, we only rely on the Synopsys tool to conduct a fully automatic place and route, which leads to similar speed results for different constraint lengths. The post-layout silicon area and power estimation (when the decoders run at 200 Mbps) results are listed in Tables III–V for $K = 7$, 8, and 9, respectively. They clearly show the effectiveness of the proposed relaxed adaptive Viterbi decoders, where the power-saving efficiency improves as the constraint length increases. Notice that TB-based decoders occupy larger silicon area than their RE-based counterparts mainly because the use of the 3-pointer even scheme largely increases the total amount of memories (almost by 3x) compared with the straightforward 1-pointer scheme.

We note that, for $K = 8$ and 9, the RE-based relaxed adaptive Viterbi decoders consume less power than their TB-based counterpart. This is because the RE-based decoders can readily leverage the reduced number of survivors to reduce the power consumption in both ACS computation and RE array; while for the TB-based decoders, since we are using a standard SRAMs other than specially customized SRAMs as discussed in Section III-B, power saving only comes from the ACS computation.

TABLE III
POST-LAYOUT AREA AND POWER ESTIMATION FOR $K = 7$

		Viterbi		Relaxed Adaptive Viterbi					
		RE	TB	RE			TB		
Core Area (# of NAND gates)		50.7K	92.2K	49.4K (-2.6%)			89.4K(-3.0%)		
Power (mW) @ 200Mbps	SNR	4dB		3dB	3.5dB	4dB	3dB	3.5dB	4dB
	Decoding Computation	25.74	24.87	13.52 (-47.5%)	12.80 (-50.3%)	11.99 (-53.4%)	13.71 (-44.9%)	13.01 (-47.7%)	12.30 (-50.5%)
	Output Generation	39.96	20.84	36.42 (-8.9%)	31.75 (-20.6%)	26.50 (-33.7%)	22.69	22.61	22.52
	Total	65.70	45.71	49.94 (-24.0%)	44.55 (-32.2%)	38.49 (-41.4%)	36.40 (-20.4%)	35.62 (-22.1%)	34.82 (-23.8%)

[4]The number in parenthesis for the RE (TB) relaxed adaptive Viterbi decoder represents the percentage of decrease against the RE (TB) Viterbi decoder.

TABLE IV
POST-LAYOUT AREA AND POWER ESTIMATION FOR $K = 8$

		Viterbi		Relaxed Adaptive Viterbi					
		RE	TB	RE			TB		
Core Area (# of NAND gates)		117.0K	195.5K	108.1K (-7.6%)			184.9K(-5.4%)		
Power (mW) @ 200Mbps	SNR	4dB		3dB	3.5dB	4dB	3dB	3.5dB	4dB
	Decoding Computation	59.70	61.83	24.31 (-59.3%)	22.69 (-62.0%)	21.03 (-64.8%)	26.99 (-56.0%)	25.11 (-59.1%)	23.17 (-62.3%)
	Output Generation	108.20	38.48	50.19 (-53.6%)	40.39 (-62.7%)	31.01 (-71.3%)	38.26	38.05	37.80
	Total	167.90	99.86	74.50 (-55.6%)	63.08 (-62.4%)	52.04 (-69.0%)	65.25 (-34.7%)	63.16 (-36.8%)	60.97 (-38.9%)

TABLE V
POST-LAYOUT AREA AND POWER ESTIMATION FOR $K = 9$

		Viterbi		Relaxed Adaptive Viterbi					
		RE	TB	RE			TB		
Core Area (# of NAND gates)		264.2K	360.7K	243.9K (-7.7%)			339.9K(-5.8%)		
Power (mW) @ 200Mbps	SNR	4dB		3dB	3.5dB	4dB	3dB	3.5dB	4dB
	Decoding Computation	129.50	136.17	40.35 (-68.8%)	37.70 (-70.9%)	34.46 (-73.4%)	43.22 (-68.3%)	40.28 (-70.4%)	36.44 (-73.2%)
	Output Generation	273.20	75.33	72.85 (-73.3%)	57.48 (-79.0%)	39.68 (-85.5%)	72.98	72.44	71.76
	Total	402.70	211.50	113.20 (-71.9%)	95.18 (-76.4%)	74.14 (-81.6%)	116.20 (-45.1%)	112.70 (-46.7%)	108.20 (-48.8%)

V. CONCLUSION

This paper presents techniques at the algorithm and VLSI architecture levels to realize high-throughput state-parallel adaptive Viterbi decoding. We developed a relaxed adaptive Viterbi algorithm that completely eliminates the global survivor path metric search operation in the conventional adaptive Viterbi algorithm, and hence is much more suitable for state-parallel decoder implementation. We further developed state-parallel relaxed adaptive Viterbi decoder architectures that can readily transform the reduced computational complexity at the algorithm level to reduced switching activities and hence power consumption at the hardware level. Supported with detailed synthesis, layout, and post-layout power estimation results, we successfully demonstrated the effectiveness of the developed techniques using convolutional code decoding as a test vehicle.

REFERENCES

[1] F. Chan and D. Haccoun, "Adaptive viterbi decoding of convolutional codes over memoryless channels," *IEEE Trans. Commun.*, vol. 45, no. 11, pp. 1389–1400, Nov. 1997.
[2] S. J. Simmons, "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. 38, no. 1, pp. 3–12, Jan. 1990.
[3] 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.
[4] 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, no. 2, pp. 514–522, Feb. 1995.
[5] 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, no. 1, pp. 52–62, Feb. 1996.
[6] M. Guo, M. Ahmad, M. Swamy, and C. Wang, "FPGA design and implementation of a low-power systolic array-based adaptive Viterbi decoder," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 2, pp. 350–365, Feb. 2005.

[7] R. Tessier, S. Swaminathan, R. Ramaswamy, D. Goeckel, and W. Burleson, "A reconfigurable, power-efficient adaptive Viterbi decoder," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 484–488, Apr. 2005.
[8] S. Swaminathan, R. Tessier, D. Goeckel, and W. Burleson, "A Dynamically Reconfigurable Adaptive Viterbi Decoder," in *Proc. of Int. ACM/SIGDA Symp. Field Programmable Gate Arrays*, Feb. 2002, pp. 227–236.
[9] F. Sun and T. Zhang, "Parallel high-throughput limited search trellis decoder VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 9, pp. 1013–1022, Sep. 2005.
[10] H.-L. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Process. Mag.*, vol. 12, no. 9, pp. 42–52, Sep. 1995.
[11] C. Rader, "Memory management in a Viterbi decoder," *IEEE Trans. Commun.*, vol. 29, no. 9, pp. 1399–1401, Sep. 1981.
[12] P. J. Black and T. H. Meng, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Apr. 1993, pp. 433–436.
[13] G. Feygin and P. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 425–429, Mar. 1993.
[14] E. Boutillon and N. Demassieux, "High speed low power architecture for memory management in a Viterbi decoder," in *Proc. IEEE Int. Symp. on Circuits Syst.*, May 1996, vol. 4, pp. 284–287.
[15] P. J. Black and T. H.-Y. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.
[16] 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.
[17] K. Kanda, H. Sadaaki, and T. Sakurai, "90% write power-saving SRAM using sense-amplifying memory cell," *IEEE J. Solid-State Circuits*, vol. 39, no. 6, pp. 927–933, Jun. 2004.
[18] M. Petrov, A. M. Obeid, A. Garcia, and M. Glesner, "A multipath high speed Viterbi decoder," in *Proc. 2003 10th IEEE Int. Conf. on Electronics, Circuits Syst. (ICECS)*, Dec. 2003, pp. 1160–1163.

Fei Sun (S'06) received the B.S. and M.S. degrees in electrical engineering from Xian Jiaotong university, China, in 2000 and 2003, respectively. He has been working toward the Ph.D. degree in electrical, computer and systems engineering department at Rensselaer Polytechnic Institute, Troy, NY, since 2003.

His research interests include VLSI architectures for communication and storage systems. Currently he is working on power efficient high throughput trellis detector architecture design for read channels.

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

Currently he is an Assistant Professor in electrical, computer and systems engineering department at Rensselaer Polytechnic Institute, Troy, NY. His current research interests include algorithm and architecture co-design for communication and data storage systems, variation-tolerant signal processing IC design, fault-tolerant system design for digital memory, and interconnect system design for hybrid CMOS/nanodevice electronic systems.