Reliable Low-Latency Viterbi Algorithm Architectures Benchmarked on ASIC and FPGA

Mehran Mozaffari Kermani, *Senior Member, IEEE*, Vineeta Singh, *Member, IEEE*, and Reza Azarderakhsh, *Member, IEEE*

***Abstract*— The Viterbi algorithm is commonly applied to a number of sensitive usage models including decoding convolutional codes used in communications such as satellite communication, cellular relay, and wireless local area networks. Moreover, the algorithm has been applied to automatic speech recognition and storage devices. In this paper, efficient error detection schemes for architectures based on low-latency, low-complexity Viterbi decoders are presented. The merit of the proposed schemes is that reliability requirements, overhead tolerance, and performance degradation limits are embedded in the structures and can be adapted accordingly. We also present three variants of recomputing with encoded operands and its modifications to detect both transient and permanent faults, coupled with signature-based schemes. The instrumented decoder architecture has been subjected to extensive error detection assessments through simulations, and application-specific integrated circuit (ASIC) [32 nm library] and field-programmable gate array (FPGA) [Xilinx Virtex-6 family] implementations for benchmark. The proposed fine-grained approaches can be utilized based on reliability objectives and performance/implementation metrics degradation tolerance.**

***Index Terms*— Error detection, look-ahead technique, recomputing with encoded operands, trellis.**

I. Introduction

THE VITERBI algorithm is an efficient method for decoding convolutional codes [1], widely used in communication systems. This algorithm is utilized for decoding the codes used in various applications including satellite communication, cellular, and radio relay. Moreover, the Viterbi decoder has practical use in implementations of high-speed (5 to 10 Gb/s) serializer-deserializers (SERDESs) which have critical latency constraints. SERDESs can be further used in local area and synchronous optical networks of 10 Gb/s. Furthermore, they are used in magnetic or optical storage systems such as hard disk drive or digital video disk [2].

The Viterbi algorithm process is similar to finding the most-likely sequence of states, resulting in sequence of observed events and, thus, boasts of high efficiency as it consists of finite number of possible states. Viterbi decoders are composed of three major components: branch metric unit (BMU), add-compare-select (ACS) unit, and survivor path memory unit (SMU). BMU generates the metrics corresponding to the binary trellis depending on the received signal, which is given as input to ACS which, then, updates the path metrics. SMU is responsible for managing the survival paths and giving out the decoded data as output. BMU and SMU units happen to be purely forward logic. ACS recursion consists of feedback loops. Therefore, the speed is limited by the iteration bound [3]. Thus, the ACS unit becomes the speed bottleneck for the system. M-step look-ahead technique can be used to break the iteration bound of the Viterbi decoder of constraint length K [4]–[10]. A look-ahead technique can combine several trellis steps into one trellis step, and if $M > K$, then throughput can be increased by pipelining the ACS architecture, which helps in solving the problem of iteration bound, and is frequently used in high-speed communication systems. Branch metric precomputation (BMP) which is in the front end of ACS is resulted due to the look-ahead technique and it dominates the overall complexity and latency for deep look-ahead architectures. BMP consists of pipelined registers between every two consecutive steps and combines binary trellis of multiple-steps into a single complex trellis of one-step. Before the saturation of the trellis, only add operation is needed. After the saturation of the trellis, add operation is followed by compare operation where the parallel paths consisting of less metrics are discarded as they are considered unnecessary.

Although Viterbi algorithm architectures are used commonly in decoding convolutional codes, in the presence of very-large-scale integration (VLSI) defects, erroneous outputs can occur which degrade the accuracy in decoding of convolutional codes. In digital systems, errors can happen through various causes including alpha particles from package decay, cosmic rays creating energetic neutrons and protons, and thermal neutrons. In advanced process technologies, errors can occur due to device shrinking, reduced power supply voltages, and higher operating frequencies which increase the probability of transient errors which can significantly affect reliability of computations. In addition, single event upsets and single event transients are generated due to cosmic rays which create energetic protons and neutrons, thermal neutrons, random noise, or signal integrity problems all resulting in device errors. As it is important to counteract

Manuscript received February 3, 2016; revised July 12, 2015; accepted September 14, 2016. Date of publication October 26, 2016; date of current version January 6, 2017. This paper was recommended by Associate Editor Y. Ha.

M. Mozaffari Kermani and V. Singh are with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: m.mozaffari@rit.edu; vxs9946@rit.edu).

R. Azarderakhsh is with the Department of Computer and Electrical Engineering and Computer Science and is an I-SENSE Fellow, Florida Atlantic University, Boca Raton, FL, USA (e-mail: razarderakhsh@fau.edu).

Digital Object Identifier 10.1109/TCSI.2016.2610187

such natural faults in order to achieve fault immunity and reliability, error detection has been an important part of a number of hardware architectures in different domains, including various arithmetic unit sub-components [11], [12].

In previous work, reliable architectures have been devised to counteract natural or malicious faults [13], e.g., cryptographic architectures immune to faults through concurrent error detection [14]–[24]. In this paper, we explore two approaches for two types of sub-parts in the Viterbi algorithm. Specifically, we note that both area/power consumption and throughput/efficiency degradations need to be minimized with respect to the proposed approaches; thus, we explore signature-based approaches resulting in acceptable efficiency at the cost of area/power consumption, and recomputing with encoded operands to achieve permanent and transient error detection. For detecting the errors in the ACS unit, we utilize three variants, i.e., recomputing with shifted operands (RESO) [25], proposed modified RESO which has slightly less fault resilience effectiveness; yet, lower induced overhead, and recomputing with rotated operands (RERO) [26]. Our architectures also include hardware redundancy techniques through signature-based detection. Specifically for the adder components, we utilize a number of variants of self-checking based on two-rail encoding. The architectures to which the schemes have been applied consist of two types of low-latency and low-complexity structures of Viterbi decoders [2] with slight modifications. We summarize the contributions of this paper as follows:

- We propose error detection methods for the modified Viterbi decoder with the consideration of objectives in terms of performance metrics and reliability. The error detection approaches along with the modifications help achieving high error coverage and through the proposed throughput improvements, performance boost can be achieved. Variants of recomputing with encoded operands on a number of architectures within the modified Viterbi decoder as well as signature-based approaches (including modified self-checking based on two-rail encoding) are presented as well. To the best of authors' knowledge, the mechanisms for making the proposed structures immune to faults have not been presented before.
- We have extensively simulated the proposed error detection architectures and the obtained results help in benchmarking the error coverage. The results of our simulations show that the reliability of the proposed architecture can be ensured.
- Finally, our proposed error detection Viterbi decoders incorporating the error detection approaches are implemented on application-specific integrated circuit (ASIC) [32 nm library] and field-programmable gate array (FPGA) [Xilinx Virtex-6 family]. The results indicate that the architectures can be reliably used. The proposed approaches can be utilized based on reliability objectives and performance/implementation metrics degradation tolerance.

The organization of this paper is as follows. In Section II, preliminaries related to the Viterbi algorithm are explained. In Section III, the proposed architectures for error detection of Viterbi algorithm are presented. Section IV presents the simulation and FPGA/ASIC implementation results. We conclude the paper in Section V.

II. Preliminaries

The preliminaries for the Viterbi decoder as well as the fault model are presented in this section.

A. Look-Ahead-Based Low-Latency Architectures

This section focuses only on branch metric computation, leaving aside the operations of compare-and-discard. An optimal approach of balanced binary grouping (BBG) [2] is taken into consideration in order to remove all redundancies which are usually responsible for longer delay and extra complexity, since various paths share common computations. Branch metrics computation is said to be carried out sequentially for a conventional Viterbi decoder. Look-ahead-based approach is a highly-efficient design approach based on the BBG scheme for a general M which provides less or equal latency, and also has much less complexity compared to other existing architectures. For constraint length K and M-step look-ahead, the execution of BMP is done in a layered manner. An M-step trellis is a bigger group consisting of $\frac{M}{K}$ sub-groups with a trellis of K-step. Thus, the total numbers of $P1$ processors needed are $\frac{M}{K}$ and each $P1$ is responsible for computing K-step trellises. Accordingly, we have the complexities and latencies of $P1$ and $P2$ as $Comp._{P1} = N(\sum_{i=2}^{k} 2^i) + N^2$, $Comp._{P2} = N^2(N-1) + N^3$, and $Lat._{P1,P2} = K$, where $N = 2^{k-1}$ is the number of trellis states. For $P1$ processors, the complexity of add operation is $N\sum_{i=2}^{k} 2^i$ and that of the "compare" operation is N^2. Similarly, for $P2$ processors, the complexity of add operation is $N^2(N-1)$ and that of the compare operation is N^3.

B. Fault Model

Transient and permanent faults both have significant chance of occurrence in VLSI architectures. Moreover, single and multiple stuck-at zero and one faults can cause erroneous outputs in such structures. We consider all of these cases, and simulate our approaches for single, few, and multiple stuck-at faults to prove their efficacy.

III. Proposed Reliable Architectures

It is well-known that in different variants of concurrent error detection, either redundancy in hardware, i.e., increase in area/power/energy consumption, e.g., through error detection codes such as hamming codes, or redundancy in time, adding negligible area overhead at the expense of higher total time (throughput and latency), is performed.

In this paper, we utilize recomputing with encoded operands, where, the operations are redone for different operands for detecting errors. During the first step, operands are applied normally. In the recomputed step, the operands are encoded and applied and after decoding, the correct results can be generated. Moreover, through signature-based schemes, we propose schemes through which both transient and permanent errors can be detected.

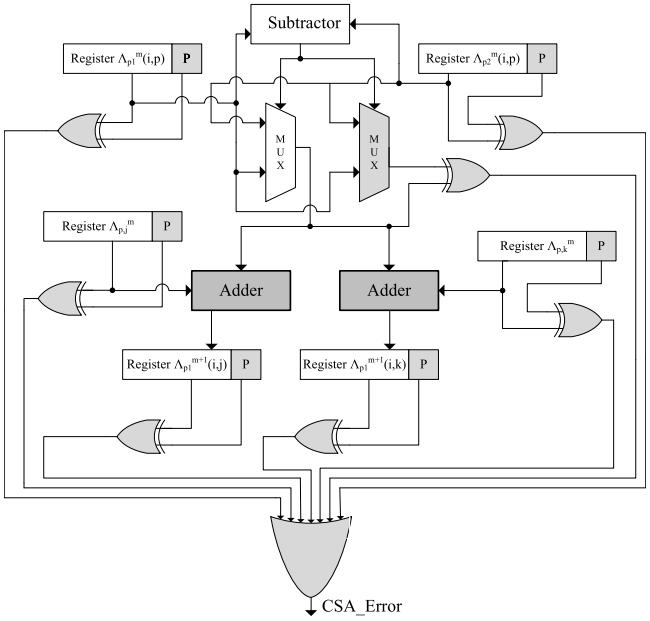

Fig. 1. The CSA signature-based error detection approach (the shaded adders are the types of the original ones with the proposed error detection schemes).

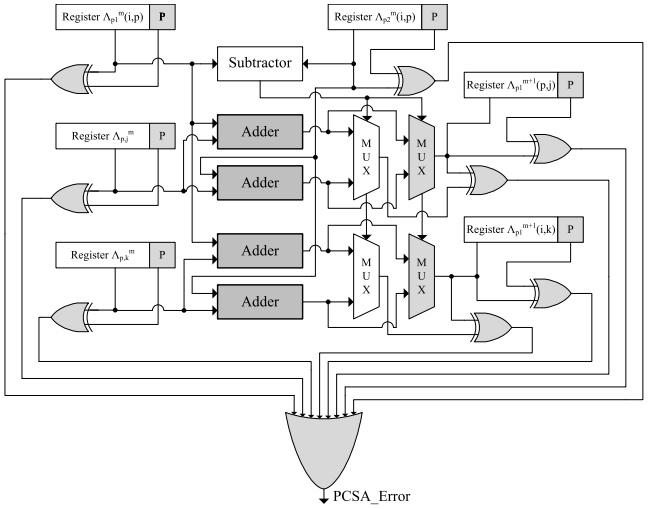

Fig. 2. Signature-based PCSA error detection (the shaded adders include the proposed error detection schemes).

A. Unified Signature-Based Scheme for CSA and PCSA Units Within BMP

In order to make the ACS structure fast, parallelization of add and compare operations within the ACS itself is done (which leads to the reduction of iteration bound delay by 50%). For achieving that, the number of states is doubled and the channel response is extended by an extra bit. For a complex trellis to have P-level parallelism, there should be 2^P parallel paths for each branch. For the initial $K-1$ steps, there is no compare operation, but for the remaining $M-K+1$ steps, the add operation is followed by a compare operation which helps in eliminating parallelism. Add and compare operations need to be performed sequentially. For this algorithm, the order of operations from add-compare is changed to compare-add and that is attributed as a carry-select-add (CSA) unit. The pre-computed CSA (PCSA) is its speed-optimized type, the details are not presented for the sake of brevity (the PCSA architecture is preferred only for large K and small M values).

We utilize signature-based prediction schemes for the CSA and PCSA units. We note that even a single stuck-at fault in such units may lead to erroneous (multi-bit) result (the error may also propagate to the circuitry which lies ahead of the affected location, with the domino effect propagated system-wise). Signatures (single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, and the like, to name a few) are employed in our proposed scheme for all the registers. Moreover, self-checking adders based on dual-rail encoding are included for the adder modules.

As shown in Figs. 1 and 2, respectively, in the CSA unit, there exists a single multiplexer whereas for the PCSA unit, the original design contains two multiplexers, for which the results of the original and the duplicated multiplexers are compared using an XOR gate whose output is connected as one of the inputs to the OR gate. The input and output registers are incorporated with additional signatures, e.g., single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, to detect faults (in figures, "P" denotes parity but it could be a chosen signature based on the overhead tolerance and reliability constraints). An OR gate for the units is required to derive the error indication flags. The OR gate raises the error indication flags (*CSA_Error* in case of the CSA unit and *PCSA_Error* in case of the PCSA unit) in case an error is detected.

For the adders included in both CSA and PCSA units, we can use self-checking adders (some previous works include [23], [24], [27]–[30]). Here, the adders are cascaded to implement a self-checking adder of arbitrary size. It consists of five two-pair two-rail checkers and also four full adders and two multiplexers are repeated n times. For the normal operation, no additional delay has resulted due to self-checking feature. The checker has two pairs of inputs driven in such a way that in the fault free scenario, the outputs are equal pairwise. This is performed using XNOR gates and appropriate connections. There are two outputs from the checker and the outputs are also in two-rail form as the inputs. Even if one of the inputs of the checker has a fault, the output is not in two-rail form and, thus, an error indication flag is raised to indicate that a fault has been incurred in the system.

The adders in both CSA and PCSA designs can also be implemented using the modified self-checking adder as shown in Fig. 3. In this variant, two n-bit ripple carry adders are used to precompute the sum bits with complemented values of carry-in, i.e., 0 and 1, and the original value of carry-in is used to select the actual sum bits. We employ this new adder [12] in the architectures and evaluate its performance and efficiency. Fig. 3 shows the design module of this variant for self-checking carry-select adder. An important modification done in this new adder is the inputs given to the two-pair two-rail checker. For carrying out the implementation for n bits, it needs $(n-2)$ AND gates, $(n+1)$ MUXes, $(n-1)$ XNOR gates, $(2n)$ full adders, and $(n-1)$ two-pair two-rail checkers.

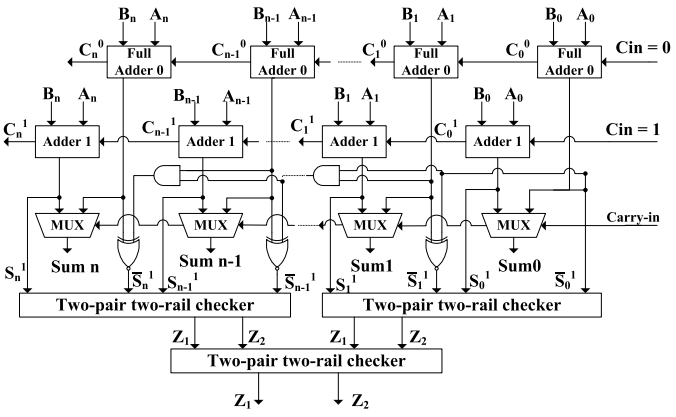

Fig. 3. A variant of self checking adder utilized in the devised approach.

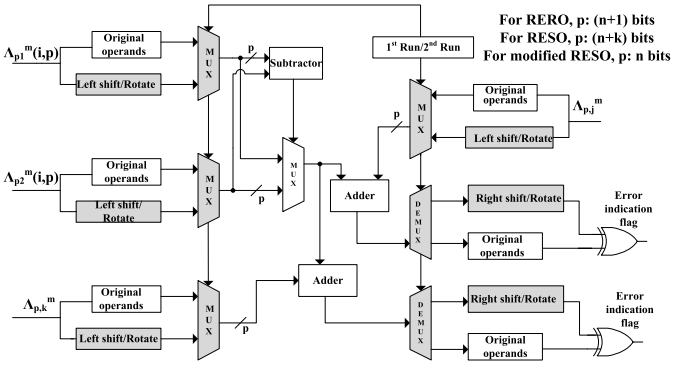

Fig. 4. Recomputing with encoded operands for CSA.

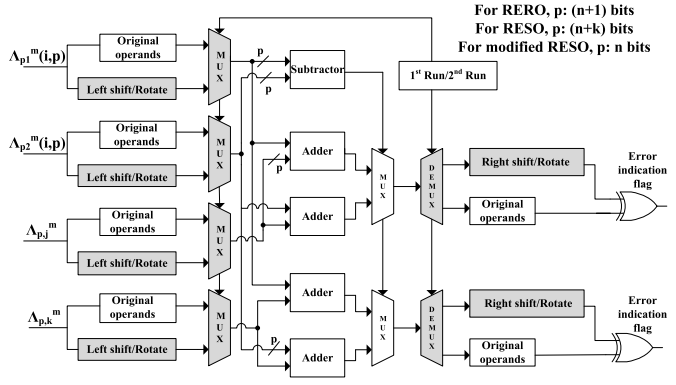

Fig. 5. PCSA error detection through recomputing with encoded operands.

B. Recomputing With Encoded Operands for CSA and PCSA

In this section, the error detection CSA and PCSA architectures are designed through recomputing with encoded operands, e.g., RERO, RESO, and variants of RESO, as shown in Figs. 4 and 5 with the locations of error detection modules shaded. Since this approach takes more number of cycles for completion, to alleviate the throughput degradation, the architecture is pipelined in the following fashion. First, pipeline registers are added to sub-pipeline the architectures, assisting in dividing the timing into sub-parts. The original operands are fed in during the first cycle. Nonetheless, during the second cycle, the second half of the circuit operates on the original operands and the first half is fed in with the rotated operands.

For the CSA and PCSA architectures in Figs. 4 and 5, we also employ RESO and a RESO variant scheme for fault diagnosis. Both CSA and PCSA units consist of four inputs, each of them are passed in its original form and in the left shifted or rotated form to one of the multiplexers. If the select lines of these multiplexers are set to the first run, the original operands are passed without any change. If these are set to second run, the second (modified, i.e., left shifted/rotated) operands are passed. For the CSA unit, the inputs are fed to the subtractor and also to the multiplexer whose select line is set by the comparator. This serves as the design of compare-select unit. The output of the multiplexer is replicated and asserted as one of the inputs to two adders included in the design. The outputs of both of the adders are the outputs of the CSA unit. These are passed through the demultiplexers and the outputs of the demultiplexers are compared using an XOR gate, and the error indication flag is raised in case of an error. For the PCSA unit, the first two inputs are fed to the comparator which acts as the select line for the two multiplexers driven by the four adders used in the design. The other two inputs in combination with the previous inputs are given to the adders. The outputs of the two multiplexers are the outputs of the PCSA unit and to ensure that they are error-free, the outputs are passed through separate demultiplexers.

We have utilized RESO which performs the recomputation step with shifted operands, i.e., all operands are shifted left or right by k bits (this method is efficient in detecting k consecutive logic errors and $k-1$ arithmetic errors). For CSA and PCSA architectures in Figs. 4 and 5, let us assume $g(x, y)$ is the result of the operation which is stored in a register. The same operation is performed again with x and y shifted by certain number of bits. This new result $g'(x, y)$ is stored and the original result $g(x, y)$ can be obtained by shifting $g'(x, y)$ in the opposite direction. Another used method in the proposed scheme is a modified version of the RESO scheme and this modification is that the bits that shift out are not preserved. This signifies that the total number of bits required for operation is only "n" bits and, hence, becomes more advantageous in terms of hardware cost than RESO and RERO methods, as pointed out in Figs. 4 and 5. In modified RESO, only $(n-k)$ LSBs of $g(x)$ are compared with the shifted $(n-k)$ LSBs of $g'(x)$. This approach is a compromise between the area/power consumption and the error coverage.

In order to execute the RERO method, we have added low hardware overhead to the initial design. RERO is used for detecting errors concurrently in the arithmetic units. Considering two n-bit rotations R and R^{-1}, suppose the input to an arithmetic function is x and $g(x)$ is the output such that $g(x) = R^{-1} \times (g(R(x)))$. The result of $g(x)$ computation happens to be the result of first run and $R^{-1} \times (g(R(x)))$ computation happens to be the second run. For both the CSA and PCSA units, we have used the RERO scheme in Figs. 4 and 5. The first challenge in RERO for in Figs. 4 and 5 is to avoid the interaction between the MSB and LSB of the original operand during the recomputation operation. The second challenge in RERO for CSA and PCSA architectures is

TABLE I
PARITY BIT OF 4-bit MEMORIES IN HEXADECIMAL FORM

$x \rightarrow, y \downarrow$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	d (1)	a (0)	9 (0)	b (1)	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	6 (0)
1	a (0)	9 (0)	b (1)	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)
2	b (1)	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)	9 (0)	1 (1)
3	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)	9 (0)	5 (0)	0 (0)
4	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)	1 (1)	5 (0)	0 (0)	2 (1)	0 (0)
5	9 (0)	9 (0)	b (1)	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	5 (0)
6	c (0)	c (0)	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)	9 (0)	2 (1)
7	8 (1)	4 (1)	6 (0)	8 (1)	9 (0)	1 (1)	5 (0)	0 (0)	2 (1)	4 (1)	5 (0)	0 (0)	1 (1)	5 (0)	0 (0)	4 (1)

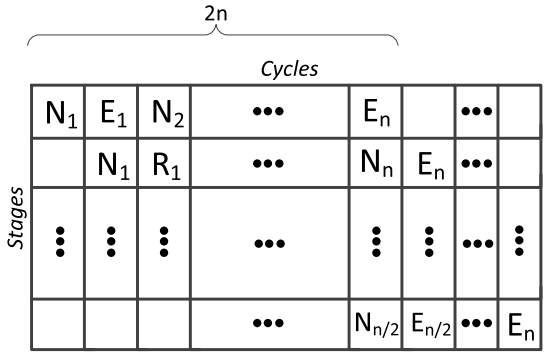

Fig. 6. Sub-pipelining for increasing the throughput.

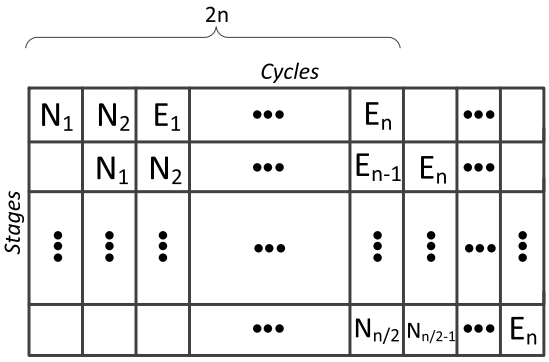

Fig. 7. Compromise in asserting the encoded operands (can be tailored based on reliability constraints).

to ensure performance enhancements through sub-pipelining to increase the frequency and alleviate the throughput overhead as part of the FPGA and ASIC implementations.

Finally, let us present a general approach for alleviating the throughput degradations of the proposed schemes. Suppose a number of pipeline registers have been placed to sub-pipeline the structures to break the timing path. Let us denote the n segments of the pipelined stages by $\Delta_1, \ldots, \Delta_n$. In a typical assertion, the original input can be first applied (to Δ_1) and in the second cycle, while the second half (Δ_2) of the architecture executes the first input, the encoded variant of the first input is fed. This trend can be scaled to n stages for normal (N) and encoded (E) operands (see Fig. 6). We have also shown in Fig. 7 an approach based on which a compromise for the assertions is performed. Depending on the requirements, one can fulfill various reliability constraints. As seen in Fig. 7, a number of cycles are considered with the normal operands shown by $N_1, \ldots, N_n$ and the encoded operands shown by $E_1, \ldots, E_n$. Let us assume that N_1 is asserted at the beginning (first stage and first cycle). We have a number of options in the second cycle, e.g., asserting the second normal operand (N_2) or the first encoded operand (encoded variant of N_1 which is E_1). Fig. 7 shows the former option as an example. In the third cycle, many options exist, among which asserting E_1 has been chosen to depict in Fig. 7. This trend continues (the sequence is $N_1, N_2, E_1, E_2, N_3, N_4, E_3, E_4, \ldots, E_{n-1}, E_n$) and after $2n$ cycles, one has $E_n, E_{n-1}, \ldots, N_{n/2}$ as the entries to various stages. Such an approach ensures lower degradation in the throughput at the expense of more area overhead and can be tailored extensively based on the overhead tolerance and the reliability requirements.

C. Signature-Based Architectures for Memories for SMU and BMU Decisions

In what follows, knowing that we need to also protect the memory units used in the CORDIC computations (for instance, storing the decisions based on the results of BMU and also the memory unites used within SMU), we present signature-based error detection for such units, for instance, parity-based and interleaved-parity-based structures. Let us use the following two examples to clarify the schemes noting that (a) the schemes are general for the needed signatures but for clarity and without being confined to just these examples, we use parities and (b) the memory size could vary, and the presented examples are for 4-bit entries with two-dimension addresses, i.e., 3-bit y and 4-bit x; however, extension to larger memories or reduction to smaller ones follow the same trend.

Example 1: We can store the predicted parities of elements in memories (memory macros on ASIC platform or block memories/pipelined distributed look-up tables on FPGAs). The parity-based scheme proposed for such architectures is based on deriving the predicted parities as shown in Table I. We modulo-2 add all bits for entries of such memories (note that the entries are randomly assumed for 4-bit memories [tabulated in hexadecimal form] to assess the details of the scheme, and x, y are the input address bits). Then, we store the result as parity bit in an extended look-up table with five-bit elements. Thus, the new, protected state would consist of 16 five-bit elements. To detect errors, this predicted parity as shown in Table I is compared with the actual parity.

Example 2: The other signature-based error detection scheme is based on interleaved parity bits that is proposed in order to protect the memories. Interleaved parity-based

TABLE II
INTERLEAVED PARITY OF 4-bit MEMORIES IN HEXADECIMAL FORM

$x \rightarrow, y \downarrow$	0	1	2	3	4	5	6	7
0	d (10)	a (00)	9 (11)	b (01)	c (11)	8 (10)	4 (01)	6 (11)
1	9 (11)	b (01)	d (10)	a (00)	9 (11)	b (01)	c (11)	8 (10)
2	c (11)	8 (10)	9 (11)	b (01)	d (10)	a (00)	9 (11)	b (01)
3	8 (10)	4 (01)	c (11)	8 (10)	9 (11)	b (01)	d (10)	a (00)
4	c (11)	8 (10)	8 (10)	4 (01)	c (11)	8 (10)	9 (11)	b (01)
5	4 (01)	6 (11)	c (11)	8 (10)	9 (11)	b (01)	d (10)	a (00)
6	9 (11)	b (01)	8 (10)	9 (11)	b (01)	d (10)	a (00)	9 (11)
7	8 (10)	4 (01)	c (11)	8 (10)	9 (11)	b (01)	d (10)	a (00)

TABLE III
NUMBER OF DETECTED FAULTS FOR SINGLE STUCK-AT FAULTS FOR RESO AND RERO "ONLY" FOR 500 000 INJECTIONS

Architecture	Detected RESO	Error coverage (%)	Detected RERO	Error coverage (%)
CSA	497,635	99.527	497,627	99.525
PCSA	497,344	99.469	497,564	99.512

TABLE IV
NUMBER OF DETECTED FAULTS FOR 2-bit, 3-bit, 4-bit, AND MULTIPLE STUCK-AT FAULTS FOR RESO AND RERO "ONLY " FOR 500 000 INJECTIONS

Architecture	Detected RESO	Error coverage (%)	Detected RERO	Error coverage (%)
CSA (2-bit)	497,836	99.567	499,578	99.915
CSA (3-bit)	498,054	99.61	499,079	99.816
CSA (4-bit)	498,666	99.733	499,103	99.82
CSA (multiple)	498,362	99.672	499,200	99.84
PCSA (2-bit)	498,482	99.697	498,993	99.79
PCSA (3-bit)	498,266	99.653	499,018	99.803
PCSA (4-bit)	498,412	99.682	499,048	99.809
PCSA (multiple)	497,957	99.591	499,039	99.808

schemes are able to detect burst faults, i.e., adjacent multiple faults. Such faults happen in both natural defects and malicious fault attacks. In this scheme, we compute the interleaved parity bits for 4-bit memories in hexadecimal form as shown in Table II (for the sake of brevity, the input sizes, i.e., the addresses to the memories are assumed to be smaller). We have derived such parities by modulo-2 addition of odd bits and even bits with each other separately. Similarly, these 2-bit interleaved parities along with 4-bit elements of each state are stored as 6-bit elements in memories.

Reinforcing Fault Detection: Using such signature-based schemes, e.g., parity-based and interleaved parity-based, there is a type of internal memory error which is not detected. Faults in the address decode circuitry which may result in accessing a wrong location may not be detected. Although such cases might not occur, if such faults are expected, we can add a separate 1-bit memory to the elements (the predicted parity bit for the correct output byte). This would detect a mismatch between the parity bit of the correct output byte and the parity bit of the incorrect output byte.

IV. BENCHMARK AND ASSESSMENTS

In what follows, we present the results of our error simulations. Then, both ASIC and FPGA implementation results are presented for benchmark.

A. Simulations for Fault Injection Models

The fault coverage of the proposed architectures has been assessed by subjecting them to a fault model which considers permanent, transient, and single/multiple-bit stuck-at faults. The proposed error detection schemes are capable of detecting both permanent and transient faults. We inject faults at different locations and monitor the error indication flags. The fault model applied for evaluating the proposed error schemes has been realized through linear feedback shift registers (LFSRs) to generate pseudo-random test patterns (the 16-bit LFSR is implemented with the polynomial $x^{16} + x^{13} + x^{11} + 1$). Nevertheless, this does not confine the coverage of the presented work to this injection approach.

For single stuck-at faults for signature-based schemes of CSA and PCSA blocks, the coverage is 100 percent (which can be analytically proved as well) and simulations are performed extensively to confirm that. In the signature-based schemes of CSA and PCSA blocks, permanent and transient faults can be detected and the blocks predicting the signatures are included in different sub-parts of the architecture.

Let us finalize this section by presenting the followings:

- We have done two simulations and derived the number of detected faults for single stuck-at faults for RESO and RERO "only" as seen in Table III.
- We have done four simulations and derived the number of detected faults for 2-bit, 3-bit, 4-bit, and multiple stuck-at faults for RESO and RERO "only" as seen in Table IV.
- We have done four simulations and derived the number of detected faults for 2-bit, 3-bit, 4-bit, and multiple stuck-at faults for RESO and RERO combined with signature-based scheme as seen in Table V.

In the following section, it is shown that such fault coverage is at the expense of acceptable overheads on ASIC and FPGA platforms.

TABLE V

NUMBER OF DETECTED FAULTS FOR 2-bit, 3-bit, 4-bit, AND MULTIPLE STUCK-AT FAULTS FOR COMBINED SCHEMES FOR 500 000 INJECTIONS

Architecture	Sig./RESO	Error coverage (%)	Sig./RERO	Error coverage (%)
CSA (2-bit)	499,998	99.9996	499,997	99.9994
CSA (3-bit)	499,999	99.9998	499,998	99.9996
CSA (4-bit)	499,997	99.9994	499,995	99.9990
CSA (multiple)	499,995	99.9990	499,998	99.9996
PCSA (2-bit)	499,998	99.9996	499,995	99.9990
PCSA (3-bit)	499,999	99.9998	499,998	99.9996
PCSA (4-bit)	499,998	99.9996	499,997	99.9994
PCSA (multiple)	499,997	99.9994	499,999	99.9998

TABLE VI

AREA, DELAY, AND POWER CONSUMPTION BENCHMARK ON ASIC FOR CSA ARCHITECTURE

Architecture	Area (μm^2)	Gate equivalent (GE)	Delay (ns)	Power (μW)	Area overhead	Delay overhead	Power overhead
CSA	486.17	319	1.24	90.64	-	-	-
CSA_RESO (+ 2 bits)	603.84	396	1.51	98.60	24.20%	21.77%	8.78%
CSA_RERO (+1 bit)	547.17	359	1.26	91.04	12.55%	1.61%	0.44%
CSA_M_RESO	488.66	320	1.26	90.89	0.51%	1.61%	0.28%

TABLE VII

PCSA AREA, DELAY, AND POWER CONSUMPTION BENCHMARK ON ASIC

Architecture	Area (μm^2)	Gate equivalent (GE)	Delay (ns)	Power (μW)	Area overhead	Delay overhead	Power overhead
PCSA	590.8	387	0.85	88.99	-	-	-
PCSA_RESO (+ 2 bits)	731.6	480	1.01	109.54	23.83%	19.39%	23.09%
PCSA_RERO (+ 1 bit)	661.7	434	0.93	100.86	12.00%	9.93%	13.34%
PCSA_M_RESO	594.7	390	0.87	89.88	0.66%	2.25%	1.00%

B. ASIC and FPGA Implementations

We present the ASIC implementation results for TSMC 32-nm library and the FPGA implementation results for Virtex-6 family (xc6vlx75t-3ff484 device) using Xilinx ISE 14.7.

For ASIC, we use Synopsys Design Compiler, and all the design constraints are set the same for fair comparison. Moreover, medium map and optimization efforts are used. The overhead results are obtained for of the area [μm^2], the NAND-gate equivalency (denoted as gate equivalent [GE] and used as the architecture area over that of a two-input NAND gate in 32 nm TSMC which is 1.524864 μm^2), the delay (ns), the power consumption (μW) at the typical chosen frequency of 50 MHz, the throughput ($Gbps$), energy (fJ), and the efficiency (which is defined as the throughput over area, i.e., $Mbps/\mu m^2$).

For FPGA, we use Xilinx ISE 14.7 for different architectures. The overhead evaluation for FPGA is obtained for of the area (in terms of number of occupied slices, knowing that slice registers and look-up tables are within), the delay (ns), the throughput ($Gbps$), and the efficiency ($Mbps$ over the number of occupied slices).

Part 1: The architectures have been designed with the design entry Verilog HDL for the original architectures as well as error detection schemes. The results of our benchmark on FPGA and ASIC are presented in Tables VI–IX.

As seen in Tables VI and VII, ASIC benchmark results for CSA and PCSA are presented for the original architectures, RESO with two bits [CSA_RESO (+2 bits) and PCSA_RESO (+2 bits)], RERO [CSA_RERO (+1 bit) and PCSA_RERO (+1 bit)], and modified variant of RESO in which no additional bit is added [CSA_M_RESO and PCSA_M_RESO]. RESO has higher overheads (still at most 24.20%) compared to RERO and modified RESO variants which have 12.55% [0.51%], 1.61% [1.61%], 0.44% [0.28%] (for area, delay, and power consumption of CSA) and 12.00% [0.66%], 9.93% [2.25%], 13.34% [1.00%] (for area, delay, and power consumption of PCSA). For signature-based CSA, we have also derived the area overhead of 17.67%, the delay overhead of 2.02%, and the power consumption overhead of 13.48%. Furthermore, for the signature-based scheme of PCSA, the area overhead of 21.49%, the delay overhead of 15.57%, and the power consumption overhead of 13.62% are achieved.

TABLE VIII

CSA BENCHMARK THROUGH XILINX VIRTEX-6 (xc6vlx75t-3ff484 DEVICE) FPGA FAMILY

Architecture	Slices	Delay (ns)	Slice over.	Delay over.
CSA	14	0.79	-	-
CSA_RESO	16	0.89	14.29%	12.52%
CSA_RERO	16	0.85	14.29%	7.46%
CSA_M_RESO	14	0.80	negligible	1.14%

TABLE IX

XILINX VIRTEX-6 FPGA IMPLEMENTATIONS FOR PCSA

Architecture	Slices	Delay (ns)	Slice over.	Delay over.
PCSA	14	0.82	-	-
PCSA_RESO	19	0.92	35.71%	12.18%
PCSA_RERO	19	0.90	35.71%	9.62%
PCSA_M_RESO	14	0.83	negligible	1.10%

Tables VIII and IX show the results of our FPGA implementations for CSA and PCSA. Similar to the ASIC results,

TABLE X
THROUGHPUT, EFFICIENCY, AND ENERGY CONSUMPTION BENCHMARK ON ASIC FOR CSA AND PCSA

Architecture	Throughput ($Gbps$)	Power (μW)	Efficiency ($\frac{Mbps}{\mu m^2}$)	Throughput deg.	Power overhead	Efficiency deg.
CSA	6.45	90.64	14.0	-	-	-
CSA_RESO (+ 2 bits)	5.30	162.8	8.7	17.83%	80.0%	37.8%
CSA_RERO (+1 bit)	6.35	153.9	11.6	1.55%	71.1%	17.1%
CSA_M_RESO	6.35	154.1	13.0	1.55%	71.5%	7.1%
PCSA	9.41	88.99	15.9	-	-	-
PCSA_RESO (+ 2 bits)	7.92	160.9	10.8	15.80%	80.1%	32.1%
PCSA_RERO (+1 bit)	8.60	152.0	13.0	8.61%	70.7%	18.2%
PCSA_M_RESO	9.19	146.4	15.5	2.33%	64.5%	2.5%

TABLE XI
XILINX VIRTEX-6 FPGA IMPLEMENTATIONS FOR THROUGHPUT AND EFFICIENCY BENCHMARK FOR CSA AND PCSA

Architecture	Thro'put ($Gbps$) [over.]	Effic. ($\frac{Mbps}{\#Slice}$) [over.]
CSA	10.1	721
CSA_RESO	8.9 [10.9%]	561 [22.2%]
CSA_RERO	9.4 [6.9%]	588 [18.4%]
CSA_M_RESO	10.0 [1.0%]	714 [2.4%]
PCSA	9.7	692
PCSA_RESO	8.7 [10.3%]	457 [33.9%]
PCSA_RERO	8.8 [9.3%]	467 [32.5%]
PCSA_M_RESO	9.6 [1.0%]	687 [0.7%]

we get lower overheads for RERO and modified RESO for the FPGA implementations, i.e., 14.29% for RERO [negligible for modified RESO] and 7.46% for RERO [1.14% for modified RESO] (for area and delay of CSA), and 35.71% for RERO [negligible for modified RESO] and 9.62% for RERO [1.10% for modified RESO] (for area and delay of PCSA). For signature-based CSA, we have also derived the area overhead of 14.89% and the delay overhead of 2.78%. Moreover, for the signature-based scheme of PCSA, the area overhead of 29.79% and the delay overhead of 1.58% are achieved.

Part 2: Tables X and XI show the results of throughput, efficiency, and power consumption for different architectures. Throughput of the designs represent performance and through the presented schemes, alleviations have been performed by one-stage pipeline on both ASIC and FPGA. Moreover, for the efficiency, which takes into account concurrently both the area bloat and the performance degradations, depending on the platform, different definitions have been used, i.e., for FPGAs, throughput over number of occupied slices, and for ASICs, throughput over the areas.

As seen in Table X, for ASIC, the throughput degradation for CSA is 1.55% for RERO and modified RESO. Moreover, we have degradations for efficiencies as 17.1% and 7.1%, respectively, for these two architectures. Finally, the power overheads are 71.1% and 80.0%, respectively. In addition, for signature-based schemes, the degradations for throughput and efficiency are 2.32% and 20.5%, respectively. For PCSA, as seen in Table XI, the ASIC benchmark shows the throughput degradation of 8.61% for RERO and 2.33% for modified RESO. Furthermore, we have degradations for efficiencies as 18.2% and 2.5%, respectively, for these two architectures. We note that higher power overheads are the expenses that we have to pay for better throughputs and efficiencies.

Finally, Table XI presents the results of the FPGA benchmarks for CSA and PCSA (for throughput and efficiency). The lowest degradations are achieved for the modified RESO architectures which make them suitable approaches for high-throughput and efficient structures.

V. CONCLUSION

In this paper, we have presented error detection architectures for the CSA and PCSA structures of low-complexity and low-latency Viterbi decoder. The proposed approaches are based on signatures and various, fine-tuned recomputing with rotated operands. The simulation results for the proposed architectures for both CSA and PCSA units show very high fault coverage (almost 100 percent) for the utilized fault model. Moreover, the ASIC and FPGA implementation results show that overheads obtained are acceptable. One may tailor the proposed architectures to have fine-tuned compromise for overhead tolerance and reliability requirements.

ACKNOWLEDGMENT

This work has been partly funded by Texas Instruments Faculty award, granted to Mehran Mozaffari-Kermani and Reza Azarderakhsh.

REFERENCES

[1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.
[2] R. Liu and K. Parhi, "Low-latency low-complexity architectures for Viterbi decoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 10, pp. 2315–2324, Oct. 2009.
[3] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.
[4] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 785–790, Aug. 1989.
[5] V. Gierenz, O. Weiss, T. Noll, I. Carew, J. Ashley, and R. Karabed, "A 550 mb/s radix-4 bit-level pipelined 16-state 0.25-μm CMOS Viterbi decoder," in *Proc. IEEE Int. Conf. Appl.-Specific Syst. Archit. Process.*, Jul. 2000, pp. 195–201.
[6] 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. 12, pp. 1877–1885, Dec. 1992.
[7] T. Gemmeke, M. Gansen, and T. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 37, no. 7, pp. 941–948, 2002.
[8] A. Yeung and J. Rabaey, "A 210 Mb/s radix-4 bit-level pipelined Viterbi decoder," in *Proc. IEEE Conf. Int. Solid-State Circuits*, Feb. 1995, pp. 88–89.

[9] 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. 504–511, Mar. 2004.

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

[11] D. Vasudevan, P. Lala, and J. Parkerson, "Self-checking carry-select adder design based on two-rail encoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 12, pp. 2696–2705, Dec. 2007.

[12] M. Akbar and J.-A. Lee, "Comments on 'self-checking carry-select adder design based on two-rail encoding'," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2212–2214, Jul. 2014.

[13] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 121–128, Jan. 2003.

[14] C.-H. Yen and B.-F. Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 720–731, Jun. 2006.

[15] T. G. Malkin, F. Standaert, and M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Workshop, Fault Diagnosis Tolerance Cryptography*, 2006, pp. 159–172.

[16] M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of Pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 12, pp. 2804–2812, Dec. 2015.

[17] M. Mozaffari Kermani and R. Azarderakhsh, "Reliable hash trees for post-quantum stateless cryptographic hash-based signatures," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT)*, Oct. 2015, pp. 103–108.

[18] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Concurrent structure-independent fault detection schemes for the advanced encryption standard," *IEEE Trans. Comput.*, vol. 59, no. 5, pp. 608–622, May 2010.

[19] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," *IEEE Trans. Ind. Electron.*, vol. 60, no. 12, pp. 5925–5932, 2013.

[20] P. Maistri and R. Leveugle, "Double-data-rate computation as a counter measure against fault analysis," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1528–1539, 2008.

[21] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A parity code based fault detection for an implementation of the advanced encryption standard," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst. (DFT)*, 2002, pp. 51–59.

[22] M. Mozaffari Kermani, R. Azarderakhsh, C. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended Euclidean-based division over GF(2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 61, no. 2, pp. 995–1003, Feb. 2014.

[23] M. Mozaffari Kermani, R. Ramadoss, and R. Azarderakhsh, "Efficient error detection architectures for CORDIC through recomputing with encoded operands," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 2154–2157.

[24] M. Mozaffari-Kermani, K. Tian, R. Azarderakhsh, and S. Bayat-Sarmadi, "Fault-resilient lightweight cryptographic block ciphers for secure embedded systems," *IEEE Embedded Syst. Lett.*, vol. 6, no. 4, pp. 89–92, Dec. 2014.

[25] J. Patel and L. Fung, "Concurrent error detection in ALUs by recomputing with shifted operands," *IEEE Trans. Comput.*, vol. C-31, no. 7, pp. 589–595, 1982.

[26] J. Li and E. Swartzlander, "Concurrent error detection in ALUs by recomputing with rotated operands," in *Proc. IEEE Int. Workshop Defect Fault Tolerance VLSI Syst.*, Nov. 1992, pp. 109–116.

[27] T. Jamil, "An introduction to complex binary number system," in *Proc. IEEE Int. Conf. Inf. Comput.*, Apr. 2011, pp. 229–232.

[28] Y. Kim and L.-S. Kim, "A low power carry select adder with reduced area," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2001, pp. 218–221.

[29] S. Mozafari, M. Fazeli, S. Hessabi, and S. Miremadi, "A low cost circuit level fault detection technique to full adder design," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Dec. 2011, pp. 446–450.

[30] F. Shih, "High performance self-checking adder for VLSI processor," in *Proc. IEEE Conf. Custom Integr. Circuits*, May 1991, pp. 15.7/1–15.7/3.

Mehran Mozaffari Kermani (S'00–M'11–SM'16) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, Canada, in 2007 and 2011, respectively. He joined the Advanced Micro Devices as a senior ASIC/layout designer, integrating sophisticated security/cryptographic capabilities into accelerated processing.

In 2012, he joined the Electrical Engineering Department, Princeton University, NJ, USA, as an NSERC postdoctoral research fellow. Currently, he is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA.

Currently, he is serving as an Associate Editor for the *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, the *ACM Transactions on Embedded Computing Systems*, the *IEEE Transactions on Circuits and Systems—Part I*, and the Guest Editor for the *IEEE Transactions on Dependable and Secure Computing* for the Special Issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations (2016 and 2017). He was the lead Guest Editor for the *IEEE/ACM Transactions on Computational Biology and Bioinformatics* and the *IEEE Transactions on Emerging Topics in Computing* for special issues on security.

He was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011 and the Texas Instruments Faculty Award (Douglas Harvey) in 2014.

Vineeta Singh received the B.E. degree in electronics and telecommunication from Pune University, India, in 2012. Later, she joined Computational Research Laboratories and designed the front end of a DDR3 Memory Controller. In 2013, she was awarded a Merit Scholarship by Tatachem Golden Jubilee Foundation; a further Scholarship was awarded by Sakal India Foundation. She received her M.Sc. degree from the Department of Electrical and Microelectronic Engineering under the supervision of Prof. Mehran Mozaffari Kermani and co-supervision of Prof. Reza Azarderakhsh at Rochester Institute of Technology, Rochester, NY, USA, in 2015. Her research interests include high-performance architectures, VLSI reliability, and reconfigurable computing.

Reza Azarderakhsh received the B.Sc. degree in electrical and electronic engineering and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, Canada, in 2011. He joined the Department of Electrical and Computer Engineering, University of Western Ontario, Canada, as a Limited Duties Instructor, in September 2011. He has been an NSERC Postdoctoral Research Fellow with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada.

Currently, he is serving as an Associate Editor for the *IEEE Transactions on Circuits and Systems—Part I*. He is the Guest Editor for the *IEEE Transactions on Dependable and Secure Computing* for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations (2016 and 2017). He is also the Guest Editor for the *IEEE Transactions on Computational Biology and Bioinformatics* for the special issue of Emerging Security Trends for Biomedical Computations, Devices, and Infrastructures (2015 and 2016). Currently, he is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA. His current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography. He was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Research Fellowship in 2012.