International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com, Volume 5, Issue 3 (May-June, 2017), PP. 96-101

RECONFIGURABLE HARDWARE DESIGN FOR HIGH SPEED VITERBI DECODER ARCHITECTURE

[1] SOUBHAGYA SASIKUMAR, [2] JUBY RAJU
[1,2] ELECTRONICS AND COMMUNICATION ENGINEERING
[1,2] MUSALIAR COLLEGE OF ENGINEERING AND TECHNOLOGY
City, Country PATHANAMTHITTA, KERALA, INDIA
[1] soubhagyasasikumar@gmail.com

Abstract — **The Viterbi algorithm is a maximum-likelihood algorithm for decoding of convolution codes used in communications such as satellite communication, cellular relay, and wireless local area networks. 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. This paper present variants of recomputing with encoded operands, to detect both transient and permanent faults, and signature-based schemes with compare select adder (CSA) unit. The adders used in the proposed one is a modified self checking adders (MSeCA).The instrumented decoder architecture has been subjected to extensive error detection assessments through simulations and field programmable gate array (FPGA) implementations for benchmark.**

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

I. INTRODUCTION

A Viterbi Decoder (VD) is employed to decode the convolutional codes, where convolutional codes are commonly used to encode digital data before transmission. This Viterbi decoder was developed by Andrew J. Viterbi in 1967 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.

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)[2].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 as shown in Fig.1[3]. BMU and SMU units happen to be purely forward logic.

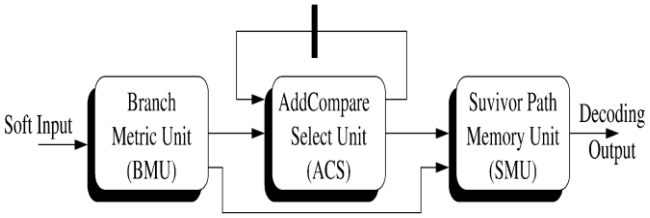

Fig. 1. Basic computation units in Viterbi decoder.

ACS recursion consists of feedback loops. Therefore, the speed is limited by the iteration bound [4]. 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 [5]. 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. 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 with modified self-checking adders based on two-rail encoding are presented as well.

• We have extensively simulated the proposed error detection architectures and the obtained results help in benchmarking the error coverage.

• Finally, our proposed error detection Viterbi decoders with modified self checking adders are implemented on field-programmable gate array (FPGA) [Xilinx Virtex-6 family].The proposed approaches can be utilized based on reliability objectives and performance/implementation metrics degradation tolerance.

II. PRELIMINARIES

A. Balanced Binary Grouping (BBG) Scheme

This section only deal with complex trellis branch metric computation without considering compare and discard operations. An optimal approach of balanced binary grouping (BBG)[2] is taken into considerations 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. When two consecutive binary-trellis steps are combined, the computation complexity is 4N, where each state has two outgoing branches and two incoming branches. In combining two complex trellis steps, the complexity can be determined. This decomposition removes all redundancy as much as possible and leads to the minimal complexity and delay. Because the trellis steps are grouped in a balanced binary way, so it refer to the optimal approach as BBG.

B. Look-Ahead- Based Low Latency Architectures

Look-ahead-based approach[9] is a highly-efficient design approach based on the BBG[2] 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 M /K sub-groups with a trellis of K-step. Thus, the total numbers of P1 processors needed are M/K and each P1 is responsible for computing K step trellises. For P1 processors, the complexity of add operation is N and that of the "compare" operation is N2. Similarly, for P2 processors, the complexity of add operation is N2(N − 1) and that of the compare operation is N3. However, the complexity of P2 is much larger than that of P1, although they have the same latency of K. Since the BBG approach is very efficient in computing equivalent branch metrics, more operations of trellis combination can be allotted into BBG-based P1 processors in order to reduce the number of P2 processors as they are expensive in terms of complexity. The trellis Steps L, which is computed in the P1 processors, has the constraint of being less than 2×K in order to make sure that the latency feature is not lost.

III. PROPOSED RELIABLE ARCHITECTURES

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[1], we propose schemes through which both transient and permanent errors can be detected.

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 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 and is preferred only for large K and small M values.

We utilize signature-based prediction schemes for the CSA and PCSA units[1]. 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, a modified self-checking adders (MSeCA) based on dual-rail encoding are included for the adder modules.

As shown in Figs.2 and 3, 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 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.

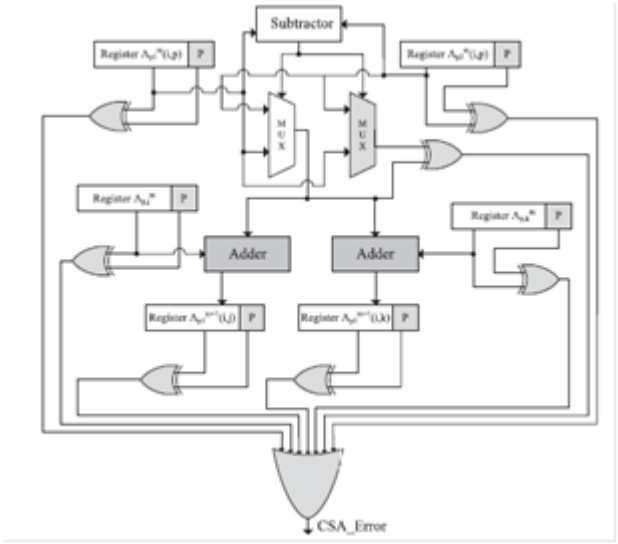

Fig.2. The CSA signature-based error detection approach (the shaded adders are the types of the original ones with the proposed error detection schemes).[1]

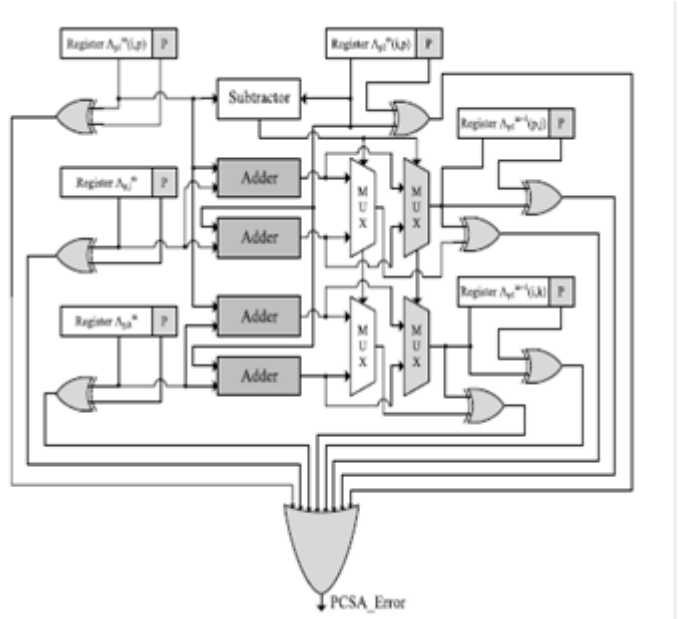

Fig. 3. Signature-based PCSA error detection (the shaded adders include the proposed error detection schemes).[1]

For the adders included in both CSA and PCSA units[1], we use a modified self-checking adders(MSecA). Here, the adders based on MUXes are cascaded to implement a self-checking adder of arbitrary size. 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 pair wise. 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. 4. In this variant, two n-bit full adders based on multiplexers are used to pre compute 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 in the architectures and evaluate its performance and efficiency. Fig.4 shows the design module of this variant for self-checking carry-select adder. An important modification done in this new adder is the full adder based multiplexer operation. For carrying out the implementation for n bits, it needs (n − 2) AND gates, (n) MUXes, (n − 1) XNOR gates, (2n) full adders, and (n − 1) two-pair two-rail checkers.
This proposed the new architecture of full adder (FA) on FPGA platform aimed with two optimization goals[6]. The first one is to optimize the FA in order to increase the speed. The second one is to develop new circuit for FA to enable a production with fewer gates. As a result, the proposed design is based on the multiplexer, and it only has two kinds of components: NOT gate and multiplexer, which makes is the design easily implemented on FPGA chip. FA is a basic cell in any unit and is so fundamental that changes to it are difficult to make. However, this cannot prevent researchers to try to increase the speed for FA. A standard FA is based on three kinds of gates such as XOR, AND and OR gates.

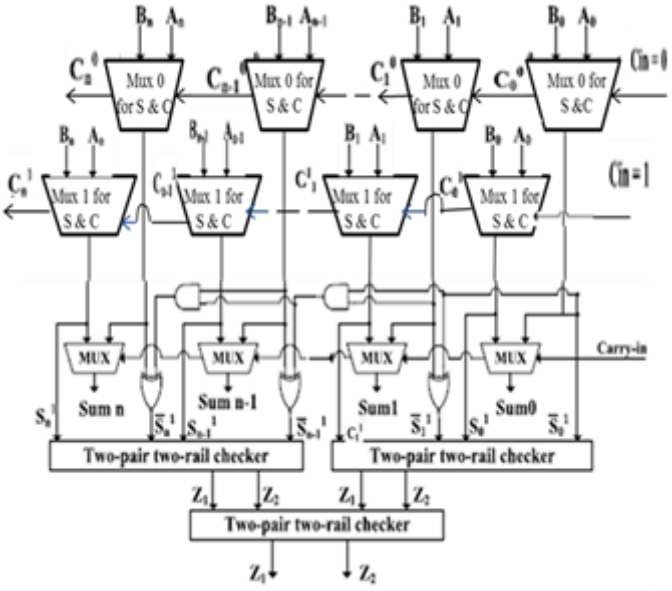

Fig.4. Modified self checking adder utilized in the devised approach[1]

Here suggest the use of only 1 or 2 kinds of gates NAND or all NOR and NOT to avoid using XOR gate with the aim to accelerate FA and reduce the number of gates. Two circuits are built for sum and carry-out based on MUX2-1. Those use MUX as a switch which is controlled to allow expected values to pass through. A circuit is built through a control of values

including input x or input y that are shown in Table.1 and Fig.5. Similarly, the circuit for carry-out of Addition are illustrated in table 2 and Fig.6.

Table.1: Truth table for Sum value [6]

Controlled by C_{in} & y			Controlled by C_{in} & x		
C_{in}	y	Sum	C_{in}	x	Sum
0	0	x	0	0	y
0	1	$\overline{x}$	0	1	$\overline{y}$
1	0	$\overline{x}$	1	0	$\overline{y}$
1	1	x	1	1	y

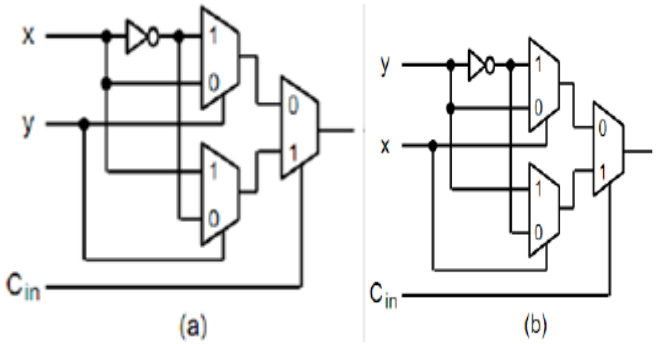

Figure 5: Circuit for Sum: (a) Controlled by Cin & y; (b) Controlled by Cin & x[6].

Table.2: Truth table for Cout value[6]

Addition		
C_{in}	x	C_{out}
0	0	0
0	1	y
1	0	y
1	1	1

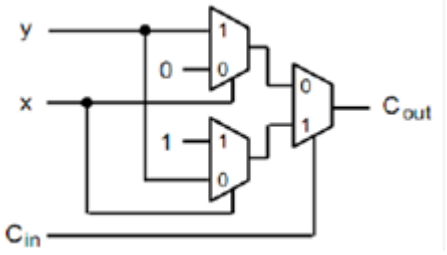

Figure 6 Carry-out flag of Addition [6]

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[1], e.g., RERO, RESO, and variants of RESO, as shown in Figs. 7 and 8 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. 7and 8, 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.

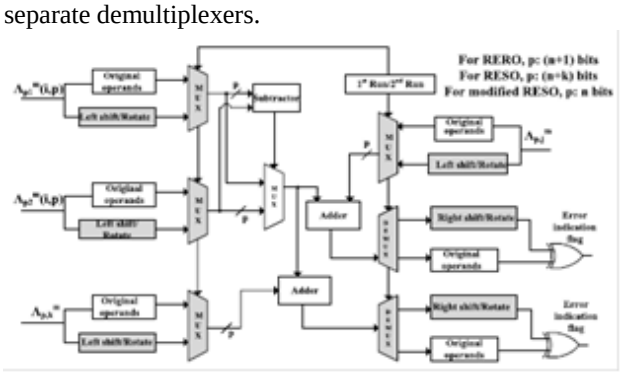

Fig. 7. Recomputing with encoded operands for CSA.[1]

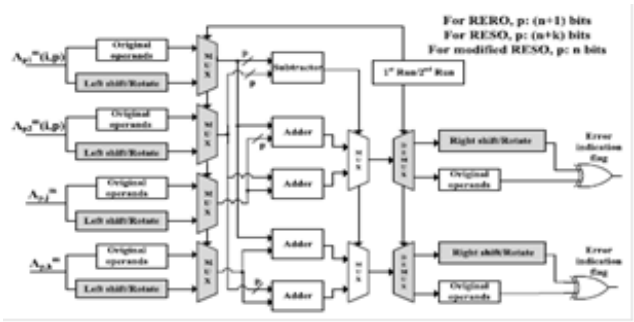

Fig.8 PCSA error detection through recomputing with encoded operands[1].

IV. IMPLEMENTATION RESULTS

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. 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 3.

Table 3.Number of detected faults for single stuck-at faults for reso and rero "only" for 500 000 injections[1]

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

For FPGA, we use Xilinx ISE 14.7 for different architectures. Table 4 shows the CSA benchmark through XILINX FPGA family. Table 5 shows Device Utilization Summary (estimated values) of modified self checking adder. The parameters indicate the number of Slices, LUTs and FF for implementing the current design.

Table 4.CSA benchmark through XILINX FPGA family[1]

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%

Table 5 Device utilization summary after Modification in self checking adder

Slice Logic Utilization	**Used**	**Available**
Number of Slice Registers	5	437600
Number of Slice LUTs	76	218800
Number used as Logic	76	218800
Slice Logic Distribution	**Used**	**Available**
Number with an unused Flip Flop	74	79
Number with an unused LUT	3	79
Number of fully used LUT-FF pairs	2	79
IO Utilization	**Used**	**Available**
Number of bonded IOBs	35	300

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 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.

REFERENCES

[1] Mehran Mozaffari Kermani and Reza Azarderakhsh, "Reliable Low-Latency Viterbi Algorithm Architectures Benchmarked on ASIC and FPGA," IEEE Transactions on Circuits and Systems I: Regular Papers,Volume: 64, Page(s): 208 – 216, October 2016.

[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. 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.

[4] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Hoboken, NJ, USA: Wiley, 1999.

[5] 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.

[6] Tranbichthuan Pham, Yi Wang, Li Renfa "Designing one-bit Full-Adder/Subtractor based on Multiplexer and LUTs architecture on FPGA," FPGA International Journal of Digital Content Technology and its Applications , April 2013 .

[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] 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.

[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.