ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Special Issue , August 2019

International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

A Novel Approach Of Viterbi Algorithm with Low – Latency for ASIC and FPGA

V. Laxmi Pavani, Ch. Gopala Krishna

[1]P.G. Student, Department of Electronics and communication engineering, Sree Vahini Institute of Science and Technology, Tiruvuru, Krishna District, AP, India.

[2]Associate professor, Department of Electronics and communication engineering, Sree Vahini Institute of Science and Technology, Tiruvuru, Krishna District, AP, India.

ABSTRACT: The Viterbi algorithm is commonly applied to a number of sensitive usage models including decoding convolution 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.

KEY WORDS: FPGA, ASIC, Virterbi algorithm,

I. INTRODUCTION

The Viterbi algorithm was introduced in 1967 as an efficient method for decoding convolution codes, widely used in communication systems. This algorithm is utilized for decoding the codes used in various applications including satellite communication, cellular, and radio relay. It has proven to be an effective solution for a lot of problems related to digital estimation. 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. It is an effective implementation of a discrete-time finite state Markov process perceived in memory less noise and optimality can be achieved by following the maximum-likelihood criteria. It helps in tracking the stochastic process state using an optimum recursive method which helps in the analysis and implementation.

A top-level architecture for Viterbi decoders is shown in Fig. 1.1. As seen in this figure, 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. The survival path is updated for all the states and is stored in the additional memory. 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; hence, its speed is limited by the iteration bound. Hence, 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.

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.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

Branch metric pre computation (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. BMP dominates the overall latency and complexity for deep look-ahead architectures. 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.

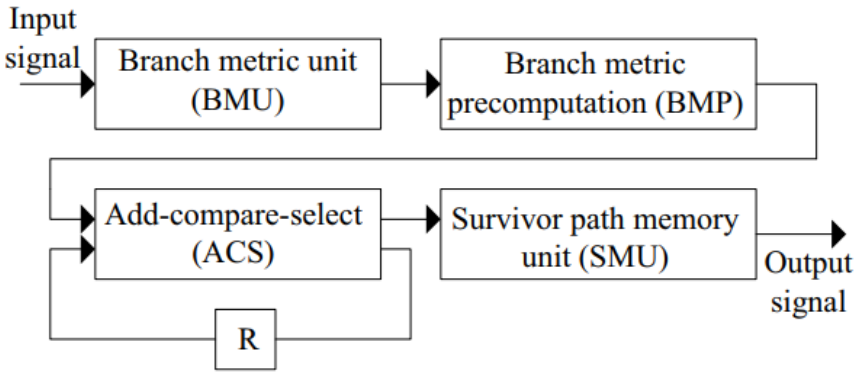

Figure 1.1: Viterbi decoder block diagram.

Although Viterbi algorithm architectures are used commonly in decoding convolution codes, in the presence of very-large-scale integration (VLSI) defects, erroneous outputs can occur which degrade the accuracy in decoding of convolution codes.

II. LITERATURE SURVEY

A. Binary Base Grouping (BBG) Approach

This section focuses only on branch metric computation, leaving aside the operations of compare-and-discard. An optimal approach of BBG 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. When two consecutive binary-trellis steps are combined, for each state, there are two incoming and two outgoing branches, and the computational complexity is $4 \times N$. As the results do not depend on the order of the trellis combination, the way the trellis steps are grouped and combined helps in determining the computational complexity. The combination in a backward nested procedure can be explained as follows. The main M-step trellises are divided into two groups consisting of m_0 and m_1 trellis steps. The binary decomposition on each subgroup goes on till it becomes a single trellis step.

The decomposition helps in removing maximum possible redundancy and, thus, helps achieve minimum delay and complexity. Finally, it can be verified that the complexities involved in the BBG approach are less as compared to the ones in the intuitive approach.

B. Look-ahead-based Low-Latency Architectures

This 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 $Comp._{P1} = N\left(\sum_{i=2}^{k} 2^i\right) + N^2$, $Comp._{P2} = N^2(N-1) + N^3$, 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 $N \sum^{k} = 2^i$ and that of the "compare" operation is N^2. Similarly, for P2 i2processors, the complexity of add operation is $N^2(N-1)$ and that of the compare operation is N^3. For both P1 and P2 processors, the latency is same, i.e., K; however, the complexity of P2 is larger than that of P1. As

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

the BBG approach is very efficient in computing the 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.

The number of groups N_g can be determined by $Ng = 2^{\lfloor \log2(\frac{M}{K}) \rfloor}$.

The overall layered structure of the Viterbi algorithm is shown in Fig. 2.1 (in this figure, i, j ∈ [1, N] and l ∈ [1, K]). As seen in this figure, within two layers (shown by Layer 1 and Layer 2 in Fig. 2.1), we have N_g steps, going through P1 and P2 processors. In each L-level P1 processor, the initial step combination is performed using the BBG approach, followed by concatenated add–compare operations executed one step at a time for the remaining L−K-step phase-II computation. In Layer 2, the outputs of P1 processors are combined for computing the final equivalent complex trellis. This figure also shows the P1 processor architecture based on the BBG algorithm. In Layer 1, although P1 leads to longer latency, as the depth of Layer 2 is reduced as well, latency penalty in not incurred.

C. Alternative to Look-ahead Approach

As the state nodes are connected pair wise, there are a total of N^2 connections, consisting of $2^{(M-K+1)}$ parallel paths. The number of parallel paths increases exponentially with respect to M, thereby, increasing the complexity. Generally, the exponential increase of parallel paths is avoided by a compare operation performed in each binary-trellis steps combination, thus, the parallel paths with less metrics are always discarded. Nonetheless, each of such add-compare operations results in a substantial amount of latency. The complexity efficiency of look-ahead depends on constraint length of Viterbi decoder. For larger constraint lengths, latency reduction is achieved at the expense of prohibitive computational complexity which limits the application of look-ahead-based architectures.

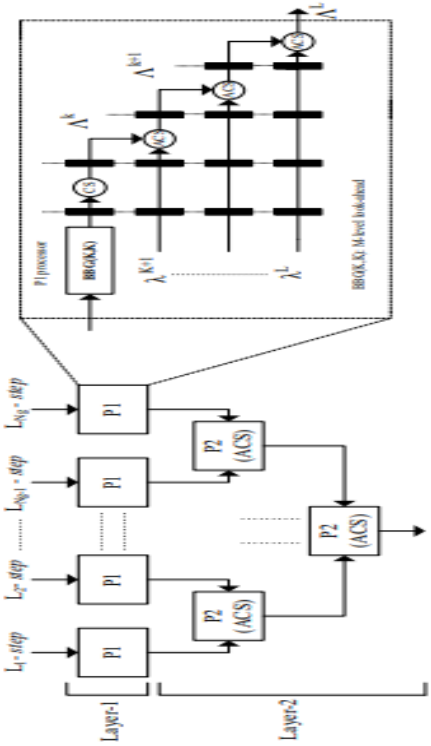

Figure 2.1: Overall layered structure including the P1 processor architecture.

III. METHODOLOGY

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Special Issue , August 2019

International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

It is outstanding that in various variations of simultaneous mistake identification, either excess in equipment, i.e., increment in territory/control/vitality utilization, e.g., through blunder location codes, for example, hamming codes, or repetition in time, including immaterial zone overhead to the detriment of higher complete time (throughput and dormancy), is performed. In this postulation, we use recomputing with encoded operands, where, the activities are revamped for various operands for recognizing blunders. During the initial step, operands are connected regularly. In the recomputed step, the operands are encoded and connected and in the wake of deciphering, the right outcomes can be created. Additionally, through mark based plans, we propose plots through which both transient and perpetual mistakes can be identified

A. Unified Signature-based Scheme for CSA and PCSA Units within BMP

The sequential branch metric computation unit is shown in Fig. 3.1. 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 2P 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, as seen in Fig. 3.1, 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 variant, 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.

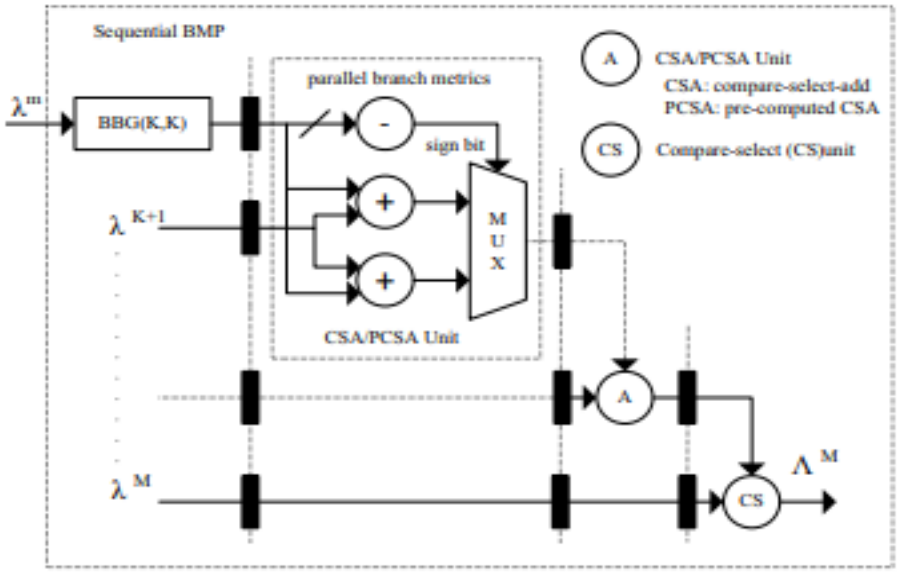

Figure 3.1.1: Sequential branch metric computation unit including CSA (PCSA) structures.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

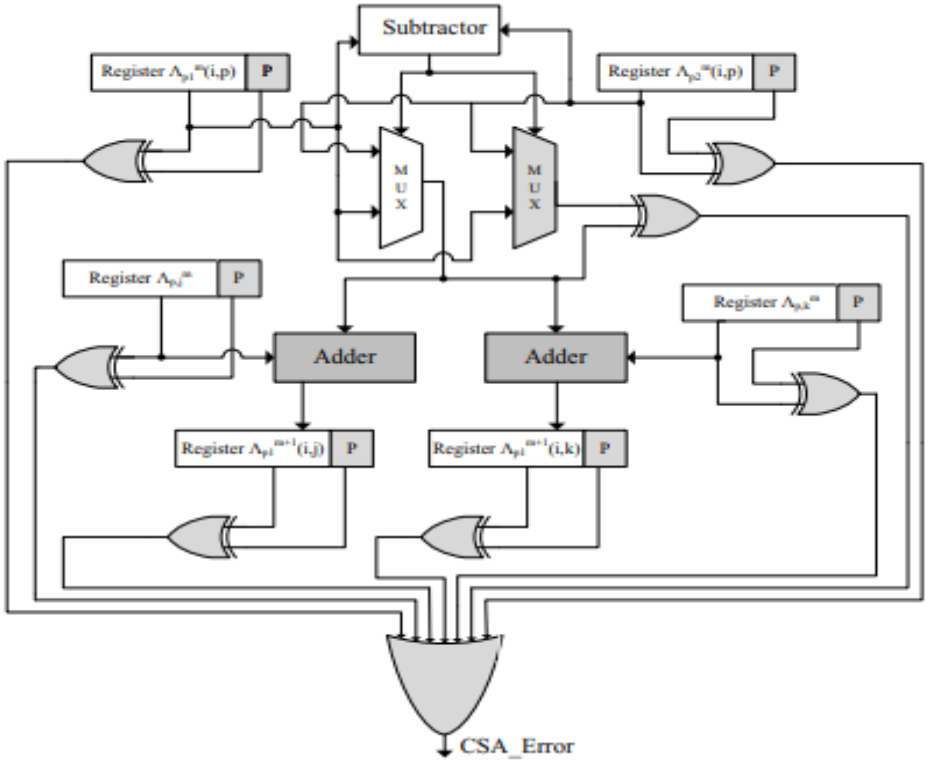

Figure 3.1.2: The CSA signature-based error detection approach

The shaded adders are variants of the original ones with the proposed error detection schemes. As shown in Figs. 3.2 and 3.3, 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 (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.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

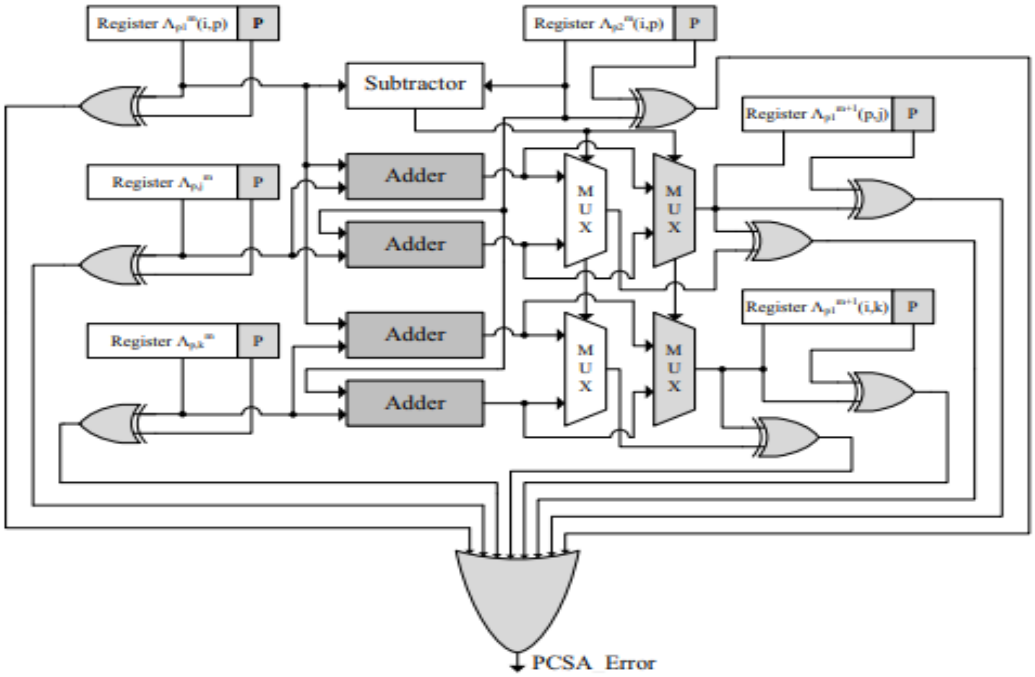

Figure 3.1.3: Signature-based PCSA error detection (the shaded adders include the proposed- error detection schemes)

The CSA signature-based error detection approach (the shaded adders are variants of the original ones with the proposed adders included in both CSA and PCSA units, we have used self-checking adders as shown in Fig. 3.4 (some previous works include [35, 36, 49–52]). As shown in this figure, 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 pair wise. This is performed using XNOR gates and appropriate connections

B. Recomputing with Encoded Operands for CSA and PCSA

In this area, the mistake identification CSA and PCSA models are structured through recomputing with encoded operands, e.g., RERO, RESO, and variations of RESO, as appeared in Figs. 3.6 and 3.7 with the areas of mistake discovery modules concealed. Since this methodology takes increasingly number of cycles for culmination, to lighten the throughput corruption, the design is pipelined in the accompanying style. To begin with, pipeline registers are added to sub-pipeline the models, helping with isolating the planning into sub-parts. The first operands are nourished in during the primary cycle. In any case, during the subsequent cycle, the second 50% of the circuit works on

For the CSA and PCSA designs in Figs. 3.6 and 3.7, we additionally utilize RESO and a RESO variation plot for deficiency finding. Both CSA and PCSA units comprise of four sources of info, every one of them are passed in its unique structure and in the left moved or turned structure to one of the multiplexers. On the off chance that the select lines of these multiplexers are set to the principal run, the first operands are passed with no change. On the off chance that these are set to second run, the second (altered, i.e., left moved/turned) operands are passed. For the CSA unit, the data sources are bolstered to the subtracter and furthermore to the multiplexer whose select line is set by the comparator. This fills in as the plan of think about select unit. The yield of the multiplexer is recreated and attested as one of the contributions to two adders incorporated into the plan. The yields of both of the adders are the yields of the CSA unit. These are gone through the demultiplexers and the yields of the demultiplexers are looked at utilizing a XOR entryway, and the blunder sign banner is brought up if there should be an occurrence of a mistake. For the PCSA unit, the initial two sources of info are nourished to the comparator which goes about as the select line for the two

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Special Issue , August 2019

International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

multiplexers driven by the four adders utilized in the structure. The other two contributions to mix with the past sources of info are given to the adders. The yields of the two multiplexers are the yields of the PCSA unit and to guarantee that they are without blunder, the yields are gone through isolated demultiplexers.

.

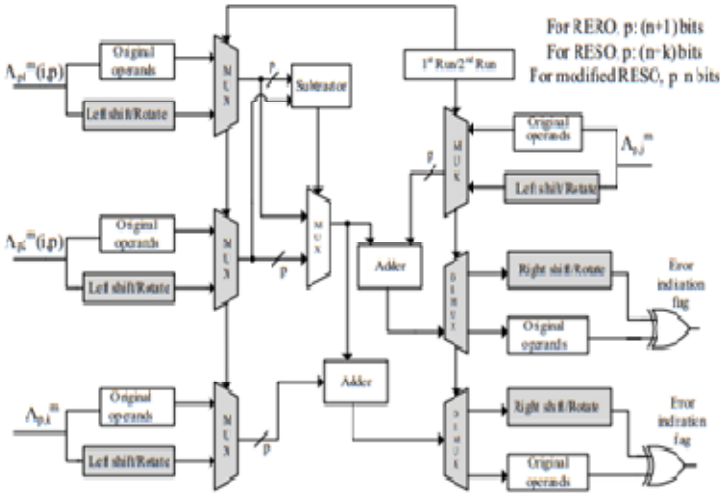

Fig3.2.1: Recomputing with encoded operands for CSA

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

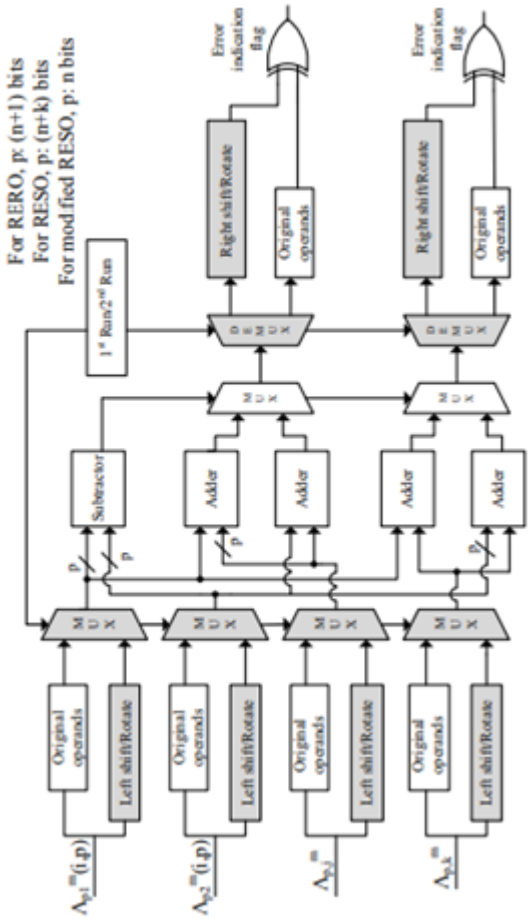

Fig3.2.2: PCSA error detection through recomputing with encoded operands

We have used RESO which plays out the recompilation venture with moved operands, i.e., all operands are moved left or appropriate by k bits (this technique is productive in recognizing k back to back rationale blunders and k − 1 number-crunching mistakes). For CSA and PCSA designs in Figs. 3.6 and 3.7, let us accept g(x, y) is the after effect of the activity which is put away in a register. A similar task is performed again with x and y moved by certain number of bits. This new outcome g ′ (x, y) is put away and the first outcome g(x, y) can be gotten by moving g ′ (x, y) the other way. Another utilized strategy in the proposed plan is an altered variant of the RESO plan and this change is that the bits that move out are not safeguarded. This implies the absolute number of bits required for task is just "n" bits and, consequently, turns out to be more invaluable as far as equipment cost than RESO and RERO strategies, as pointed out in Figs. 3.6 and 3.7.

In changed RESO, just (n−k) LSBs of g(x) are contrasted and the moved (n−k) LSBs of g ′ (x). This methodology is a trade off between the territory/control utilization and the mistake inclusion. So as to execute the RERO technique, we have added low equipment overhead to the underlying structure. RERO is utilized for recognizing blunders simultaneously in the number juggling units. Considering two n-bit pivots R and R −1 , assume the contribution to a number juggling capacity is x and g(x) is the yield with the end goal that g(x) = R −1 × (g(R(x))). The consequence of g(x) calculation happens to be the after effect of first run and R −1 × (g(R(x))) calculation happens to be the subsequent run. For both the CSA and PCSA units, we have utilized the RERO conspire in Figs. 3.6 and 3.7.

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

The main test in RERO for in Figs. 3.6 and 3.7 is to maintain a strategic distance from the association between the MSB and LSB of the first operand during the recompilation task. The second test in RERO for CSA and PCSA models is to guarantee execution upgrades through sub-pipelining to expand the recurrence and mitigate the throughput overhead as a feature of the FPGA and ASIC usage. At last, let us present a general methodology for reducing the throughput corruptions of the proposed plans. Assume various pipeline registers have been put to sub-pipeline the structures to break the planning way. Give us a chance to mean the n portions of the pipelined organizes by Δ1-Δn. In a run of the mill statement, the first information can be first connected (to Δ1) and in the subsequent cycle, while the subsequent half (Δ2) of the engineering executes the principal input, the encoded variation of the primary information is encouraged. This pattern can be scaled to n stages for typical (N) and encoded (E) operands.

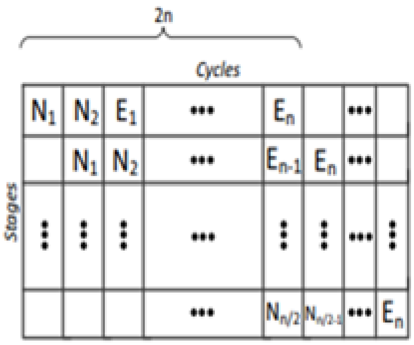

Fig3.2.3: Compromise in asserting the encoded operands (can be tailored based on reliability constraints)

We have appeared in Fig. 3.8 a methodology dependent on which a trade off for the declarations is performed. Contingent upon the prerequisites, one can satisfy different unwavering quality requirements. As found in Fig. 3.8, various cycles are considered with the typical operands appeared by N1 −Nn and the encoded operands appeared by E1 −En. Give us a chance to accept that N1 is attested toward the start (first stage and first cycle). We have various choices in the subsequent cycle, e.g., stating the second typical operand (N2) or the first encoded operand (encoded variation of N1 which is E1). Fig. 3.8 demonstrates the previous choice for instance. In the third cycle, numerous choices exist, among which stating E1 has been portrayed in Fig. 10. This pattern proceeds and after 2n cycles, one has En,En−1,...,Nn/2 as the passages to different stages. Such a methodology guarantees lower debasement in the throughput to the detriment of more zone overhead and can be customized widely dependent on the overhead resilience and the unwavering quality necessities.

IV. EXPERIMENTAL RESULTS

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	91	16640	0%
Number of Slice Flip Flops	112	33280	0%
Number of 4 input LUTs	150	33280	0%
Number of bonded IOBs	69	309	22%
Number of GCLKs	1	24	4%

Fig 4.1: Design summary

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Special Issue , August 2019

International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

```
================================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'Clock'
  Total number of paths / destination ports: 6272 / 32
--------------------------------------------------------------------------------
Offset:              12.868ns (Levels of Logic = 6)
  Source:            OB_0 (FF)
  Destination:       YC1<7> (PAD)
  Source Clock:      Clock rising

  Data Path: OB_0 to YC1<7>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               5   0.591   0.776  OB_0 (OB_0)
     LUT2:I0->O            1   0.648   0.452  SC/Mxor_S0_0_xo<0>21 (N111)
     LUT4:I2->O            1   0.648   0.563  EAB/e026 (EAB/e026)
     LUT4:I0->O           36   0.648   1.406  EAB/e0206 (e0)
     LUT4:I0->O            8   0.648   0.900  SEC/Mmux_YC31211 (SEC/N2)
     LUT4:I0->O            1   0.648   0.420  SEC/Mmux_YC39 (YC3_2_OBUF)
     OBUF:I->O                 4.520          YC3_2_OBUF (YC3<2>)
    ----------------------------------------
    Total                     12.868ns (8.351ns logic, 4.517ns route)
                                       (64.9% logic, 35.1% route)

================================================================================
```

Fig 4.2: TIME SUMMARY

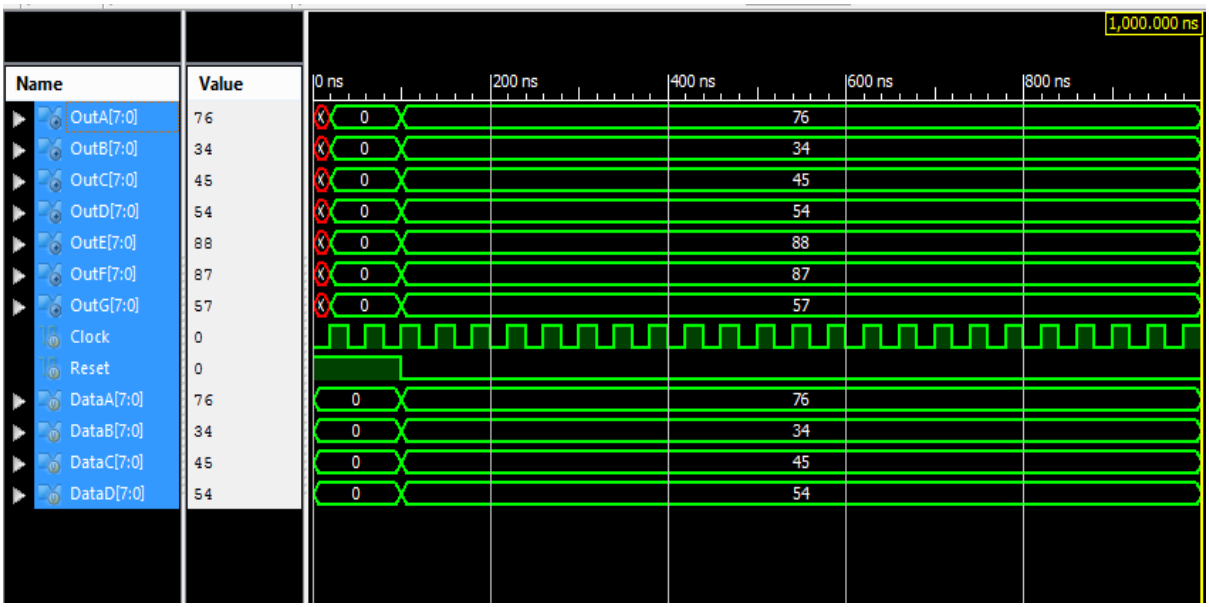

Fig 4.3: ENCODER OUTPUT

ISSN: 2350-0328
International Journal of Advanced Research in Science, Engineering and Technology
Vol. 6, Special Issue , August 2019
International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

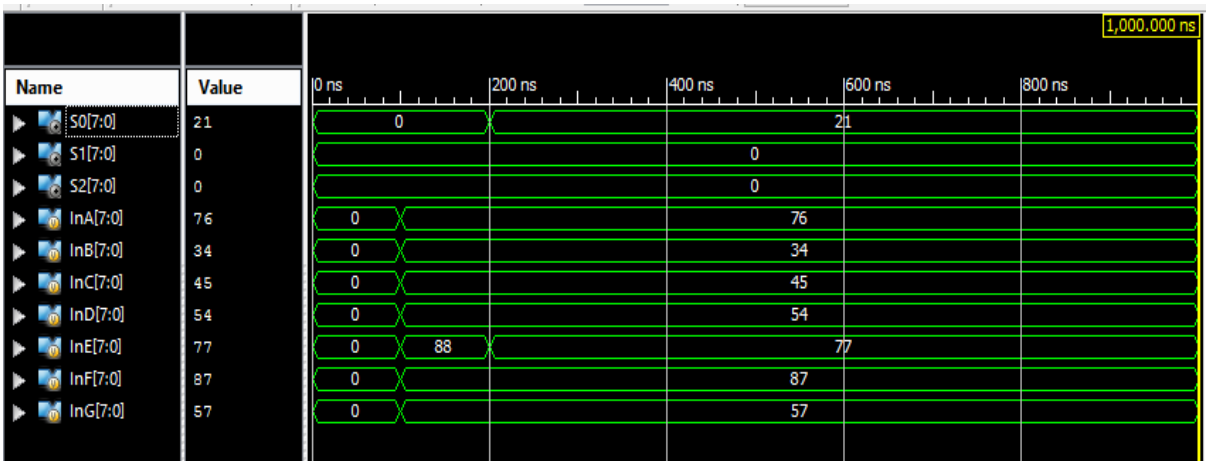

Fig 4.4: BMU OUTPUT

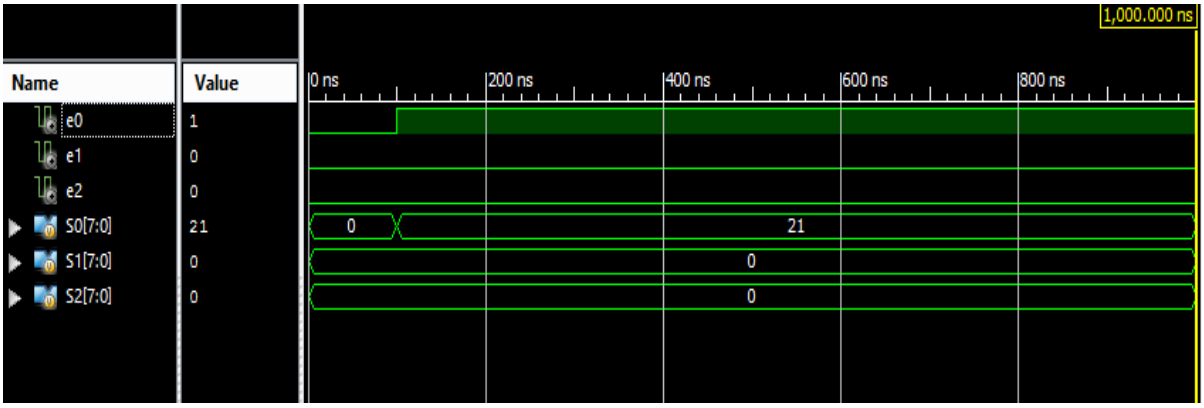

Fig 4.5: ACS OUTPUT

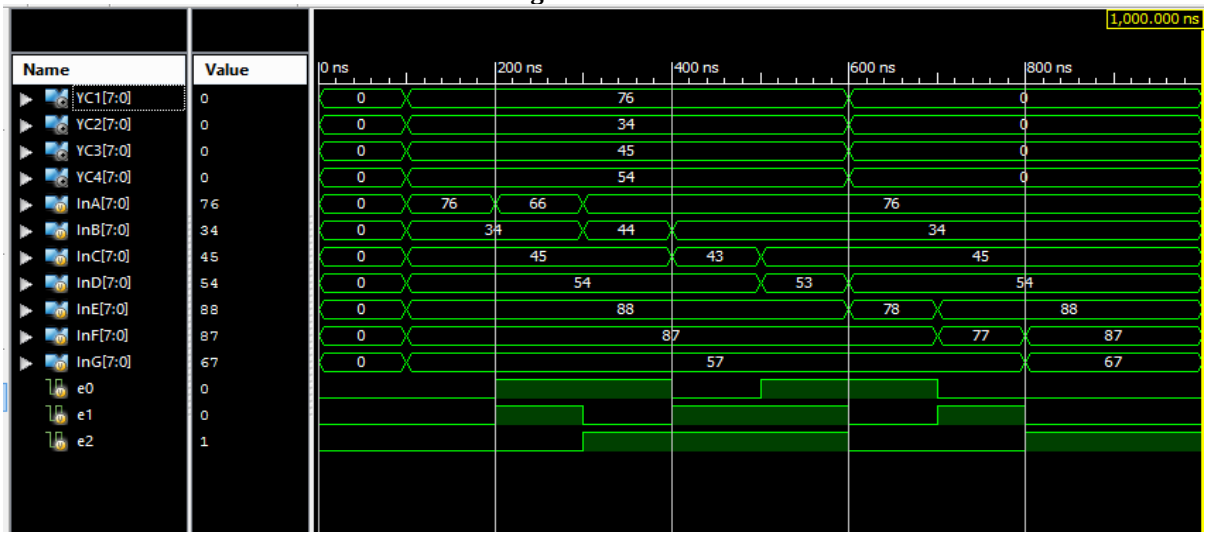

Fig 4.6: PCSA OUTPUT

ISSN: 2350-0328

International Journal of Advanced Research in Science, Engineering and Technology

Vol. 6, Special Issue , August 2019

International Conference on Recent Advances in Science, Engineering, Technology and Management at Sree Vahini Institute of Science and Technology-Tiruvuru, Krishna Dist, A.P

V. CONCLUSION AND FUTURE SCOPE

In this thesis, we presented fault diagnosis models for the CSA and PCSA units of low complexity and low-latency Viterbi decoder. The simulation results for the proposed methods of RESO, RERO, modified RESO, parity and self-checking adder based designs for both CSA and PCSA units show very high fault coverage (almost 100 percent) for the randomly distributed injected faults. The proposed architectures have been successfully implemented on Xilinx Virtex-6 Family and also by using the 32nm library using Synopsys Design Compiler for the ASIC implementation. Also, the ASIC and FPGA implementation results show that overheads obtained are acceptable. Thus the proposed models are reliable and efficient.

This thesis work focussed on performing the fault detection on the CSA unit and the PCSA unit. The work can be extended by performing fault detection for the different binary-trellis groups using the parity registers and duplicating the adders. Recomputing with encoded operands and unified signature-based scheme were used to detect faults in this work. In future, the proposed architectures can be tested with other fault detection techniques like off-line error detection schemes and roving fault detection method.

REFERENCES

[1] Massoud Pedram, “Power minimization in ic design: Principles and applications,” ACM Trans. Des. Autom. Electron. Syst., vol. 1, no. 1, pp. 3–56, Jan. 1996.
[2] Qing Wu, M. Pedram, and Xunwei Wu, “Clock-gating and its application to low power design of sequential circuits,” Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, vol. 47, no. 3, pp. 415–420, Mar 2000.
[3] G.E. Tellez, A. Farrahi, and M. Sarrafzadeh, “Activity-driven clock design for low power circuits,” in Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on, Nov 1995, pp. 62–65.
[4] E. Lee and A. Sangiovanni-Vincentelli, “Comparing models of computation,” in Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design. IEEE Computer Society, 1997, pp. 234–241.
[5] Gilles Kahn, “The Semantics of Simple Language for Parallel Programming,” in IFIP Congress, 1974, pp. 471–475.
[6] Edward A. Lee and David G. Messerschmitt, “Static scheduling of synchronous data flow programs for digital signal processing,” IEEE Trans. Comput., vol. 36, no. 1, pp. 24–35, 1987.
[7] E.A. Lee and T.M. Parks, “Dataflow process networks,” Proceedings of the IEEE, vol. 83, no. 5, pp. 773 –801, may 1995.
[8] Syed Suhaib, Deepak Mathaikutty, and Sandeep Shukla, “Dataflow architectures for GALS,” Electronic Notes in Theoretical Computer Science, vol. 200, no. 1, pp. 33–50, 2008.
[9] Tzyh-Yung Wuu and Sarma B. K. Vrudhula, “Synthesis of asynchronous systems from data flow specification,” Research Report ISI/RR-93-366, University of Southern California, Information Sciences Institute, Dec 1993.
[10] Behnam Ghavami and Hossein Pedram, “High performance asynchronous design flow using a novel static performance analysis method,” Comput. Electr. Eng., vol. 35, no. 6, pp. 920–941, Nov. 2009.
[11] S.C. Brunet, E. Bezati, C. Alberti, M. Mattavelli, E. Amaldi, and J.W. Janneck, “Partitioning and optimization of high level stream applications for multi clock domain architectures,” in Signal Processing Systems (SiPS), 2013 IEEE Workshop on, Oct 2013, pp. 177–182.
[12] Simone Casale-Brunet, Analysis and optimization of dynamic dataflow programs, Ph.D. thesis, STI, Lausanne, 2015.
[13] Endri Bezati, High-level synthesis of dataflow programs for heterogeneous platforms, Ph.D. thesis, STI, Lausanne, 2015.
[14] S. Casale-Brunet, M. Mattavelli, and J.W. Janneck, “Buffer optimization based on critical path analysis of a dataflow program design,” in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, May 2013, pp. 1384–1387.
[15] Xilinx, Analysis of Power Savings from Intelligent Clock Gating, August 2012, XAPP790.
[16] “Open RVC-CAL Applications,” 2014, http://github.com/orcc/orc-apps, accessed 25-February-2014]. [17] M. Canale, S. Casale-Brunet, E. Bezati, M. Mattavelli, and J. Janneck, “Dataflow programs analysis and optimization using model predictive control techniques,” Journal of Signal Processing Systems, pp. 1–11, 2015.