Efficient Fault Tolerant Parallel Filter Using BCH Codes with FPGA

Priyanka Patel[1], Prof. Tarun Verma[2]

[1]*Research Scholar,* [2]*Assistant Professor*

Department of Electronics and Communication Engineering, LNCT, Bhopal

Abstract - Signal processing is the bone of every digital system dealing with every field equipped with computing and logic circuits for controlling applications. The digital system is prone to several problems like faults, errors and mismatches, and for that fault tolerant filter techniques. The fault tolerant structures are designed based on different parallel architectures to reduce the errors introduced by circuit due to faults. The several error correction codes are adopted for getting better performance. This paper deals with the parallel architecture of parallel FIR filter using Bose - Chaudhuri - Hocquenghem (BCH) Codes. BCH Codes are helps to create large class of multiple random error corrections, it is a kind of cyclic codes. The design is implemented on FPGA devices. The proposed methodology has better reliable architecture of fault tolerant architecture as shown in the synthesis outcomes.

Keywords - BCH Codes, Error Corrections, FIR filter, Fault Tolerant, FPGA.

I. INTRODUCTION

The demand for processing power is increasing steadily. In many application fields, there can never be enough computing power. Simulations in the field of engineering, like virtual crash tests, or in the field of bioinformatics, as protein folding, are examples for applications that require enormous computing power. In any case, significantly customer PCs keep on demanding for more processing power. Moore's Law, foreseeing that the execution of chip duplicates about at regular intervals, has ended up being valid previously, and will in all likelihood remain valid for the not so distant future. One contributing factor to this performance increase is technological improvements. However, the direct influence of technology on computing performance is limited. Architectural improvements are another main source for sustained performance improvements.

In the past, single processor performance has been in the main focus for computer architecture. But even in this case, the exploitation of parallelism at instruction level is a key element. As instruction level parallelism is limited in single processor applications, further performance increases can only be achieved by exploiting parallelism at the higher levels of thread or process parallelism. As an outcome, current "processors" fuse various processor centers that together frame a solitary shared memory multiprocessor. While the architecture of the processor cores does not fundamentally differ from the architecture of single processors, architectural research must optimize communication among the processors.

In large parallel systems, which are typically message-passing multicomputer, a network interface controller connects the individual nodes to the network. Traditionally, the system interface controller is associated with its home node like each other information/yield device over a hierarchy of importance of peripheral interconnects. While this is a suitable answer for moderate devices like hard plates, it has turned into a huge bottleneck for system interface controllers (NIC) and coprocessor devices like field-programmable gate arrays (FPGA).

Additionally, the classical assumption that a computing node consists of a single processor with memory and I/O components is outdated. Multi-core processors have turned every computing node into a small-scale shared memory system. The pattern towards higher parallelism is self-evident: dual core processors are standard notwithstanding for personal computers, and every single real merchant are as of now presenting four or eight center processors. Inquire about models of multi-attachment frameworks include up to 80 centers on a single kick the bucket. Today's network interface architectures do not consider this fact sufficiently.

II. THEORY OF FAULT TOLERANCE

High reliability is needed in many signal processing applications to ensure continuous operation and to check the integrity of results. High reliability is needed in life critical applications, such as aircraft guidance systems or in medical equipment, where failures can jeopardize human lives, or in remote applications, such as satellites or underwater acoustic monitors, where repair is impossible or prohibitively expensive. Robustness is also needed in systems that must operate in hazardous environments, such as military equipment, or in spacecraft that must be protected against radiation. In all of these applications there is a high cost of failure, and reliability is of great importance.

The complexity of signal processing algorithms has been steadily increasing due to the availability of special purpose, high-speed signal processors. Many algorithms that were once too computationally intensive, are now implemented in real time by multiprocessor systems. In these systems, the large amount of hardware increases the likelihood of a failure occurring, and makes reliable operation difficult.

It is impossible to guarantee that components of a system will never fail. Instead, failures should be anticipated, and systems designed to tolerate failures gracefully. This design methodology is known as fault-tolerant computing and it received considerable attention by early computer designers because of the unreliability of existing components. After the development of integrated circuits, which were several orders of magnitude more reliable, fault-tolerance became a secondary issue. Attention was focused on developing faster, more complex circuits, and as a result, circuit densities grew exponentially. In many areas, however, semiconductor reliability has not kept pace with the level of integration, and fault-tolerance is becoming a major issue again.

A. Modular Redundancy

In order for a system to be fault-tolerant, it must contain some form of redundancy. By redundancy we mean additional states that arise during faults and that are used to detect and correct errors. Without redundancy, it is impossible for a system to be fault-tolerant since it is unable to distinguish between valid and invalid internal states. Utilizing redundancy is in contrast to the goal of eliminating as much redundancy as possible. Redundancy generally increases the complexity of a system and leads to increased cost.

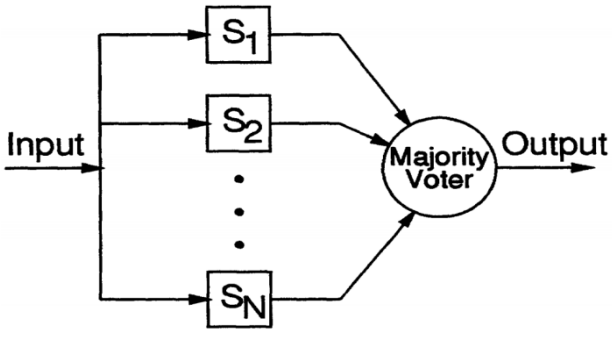

Figure 2.1 Example of N-modular redundancy used to protect system S.

The traditional method of adding redundancy and fault-tolerance to a system is through modular redundancy [7]. This is a system-level approach in which several copies of the system operate in parallel, using the same input. Their outputs are compared with voter circuitry and will agree if no errors have occurred. Otherwise, if the outputs are not identical, then an error has occurred and the correct result may be determined using a majority vote. A system, S, protected by modular redundancy is shown in Figure 2.1. Modular redundancy (MR) is the most widely used fault-tolerance technique. This is because it can be used to protect any system and since it decouples system and fault- tolerance design.

B. Coding Theory

Systems that tolerate failures have been of interest since the 1940's when computational engines were constructed from relays. Fault detection provides no tolerance to faults, but gives warning when they occur. If the dominant form of faults is transient/intermittent, recovery can be initiated by a retry invoked from a previous checkpoint in the system at whose time the system state was known to be good. Design errors, whether in hardware or software, are those caused by improper translation of a concept into an operational realization. The three major axes of the space of fault-tolerant designs are: system application, system structure, and fault-tolerant technique employed. The most stringent requirement for fault tolerance is in real-time control systems, where faulty computation could jeopardize human life or have high economic impact. Computations must not only be correct, but recovery time from faults must be minimized. Specially designed hardware is employed with concurrent error detection so that incorrect data never leaves the faulty module. The more redundancy we add, the more reliably we can detect and correct errors but the less efficient we become at transmitting the source data. Figure 2.2 demonstrated the typical diagram of system with error correcting code.

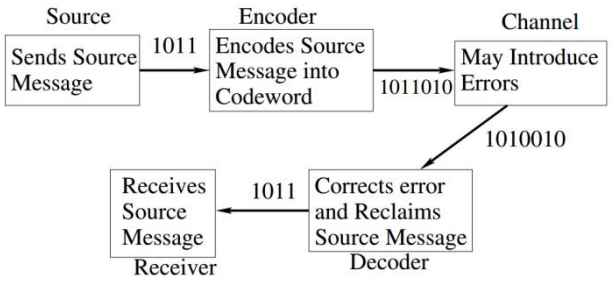

Figure 2.2 Typical Diagram of a System with an Error Correcting Code.

Forward recovery attempts to restore the system by finding a new state from which the system can continue operation. Backward recovery attempts to recover the

system by rolling back the system to a previously saved state, assuming that the fault manifested itself after the saved state. Forward error recovery, which produces correct results through continuation of normal processing, is usually highly application-dependent.

Backward recovery techniques require some redundant process and state information to be recorded as computations progress. Error detection and correction codes have proven very effective for regular logic such as memories and memory chips have built-in support for error detection and correcting codes.

III. PROPOSED ARCHITECTURE

This paper deals with the parallel architecture of parallel FIR filter using Bose - Chaudhuri - Hocquenghem (BCH) Codes. BCH Codes are helps to create large class of multiple random error corrections; it is a kind of cyclic codes. The design is implemented on FPGA devices. The proposed methodology has better reliable architecture of fault tolerant architecture as shown in the synthesis outcomes.

BCH is genuinely clear, playing out the decoding steps is a great deal more intricate (Zambelli et al. 2012). System planners must adjust the high multifaceted nature of BCH decoders with their general system necessities (Strukov 2006). The decoders must give high throughput, either by running at high clock speeds or by executing bit parallel operation. The most extreme clock speed of the decoder is constrained by the procedure innovation and the unpredictability of the decoder. Additionally, adding bit-parallel operation expands the range of the decoder and makes it more hard to accomplish high clock speeds. Constrained accessible area for the decoder can likewise confine the quantity of mistakes that can be amended.

By building up a more area effective BCH decoder, a few potential outcomes open up other than just lessening area. The area savings can be utilized to add bit-parallel operation to enhance throughput. Then again the decoder could be intended to right more blunders developing the helpful existence of blaze memory or expanding the bit-rate of a communication channel.

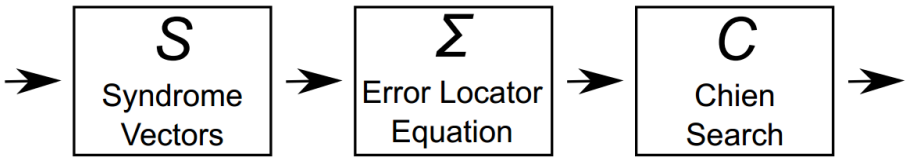

Figure 4. 1 Basic BCH decoder structure.

A common BCH decoder usage is basically a 3-arrange pipeline as appeared in figure 4.1. The three phases of the pipeline are disorder computation, producing the mistake locator polynomial, and finding the underlying foundations of the blunder locator polynomial (Hong and Vetterli 1995). Every pipeline arrange works at the same time and freely. Data is passed between the phases when the present stage is finished and the following stage is ready to get the data. This pipelined setup permits the decoder to work on 3 codes all the while. The primary stage, disorder figuring is comparable in design to encoding and at comparable cost. A basic rationale circuit known as a Linear Feedback Shift Register (LFSR) is ordinarily utilized for disorder figuring. As LFSRs are utilized as a part of encoding and disorder estimation, work has gone to streamline fast piece parallel LFSR operation for BCH.

Computing the error locator polynomial is performed successive approximation utilizing the Berlekamp-Massey algorithm. The execution of the calculation requires numerous multipliers and dividers, and expends a huge part of the decoder. General work into advanced Berlekamp-Massey usage has been done and also the sharing of Berlekamp-Massey units between BCH channels.

BCH is a block based error correction code implying that it works on a square ofbits at time (Bose and Ray-Chaudhuri 1960). It changes the info data by adding exceptionally figured excess check bits to shape a codeword. The suitable code can be chosen for various bits to be rectified and a picked square size. Bigger square sizes have bring down capacity overhead, however higher algorithmic multifaceted nature. This gives BCH various advantages, including:

- Configurability for number of bits to be corrected.
- Scales to different word sizes.
- Optimal algebraic method for decoding.
- No error floor.
- Original data embedded in codeword.

Every codeword inside the code is developed with the end goal that it is a base Hamming separation far from some other codeword. The Hamming separation, dmin is dictated by the quantity of bits that must be changed inside a legitimate codeword to change it into another

substantial codeword. The number ofbit mistakes that can be recognized is in this manner one not as much as the Hamming separation. Figure 3 demonstrates the structure of a BCH codeword, including the message and the repetitive ECC data that is added to shape the codeword.

The functionality of the decoder is to figure out which valid codeword received codeword most nearly represents. In the event that a codeword gets enough bit errors to cross half or a greater amount of the Hamming separation between two codewords, it will be erroneously distinguished. Hence the number that can be corrected, t, is identified with the base Hamming separation by the accompanying connection:

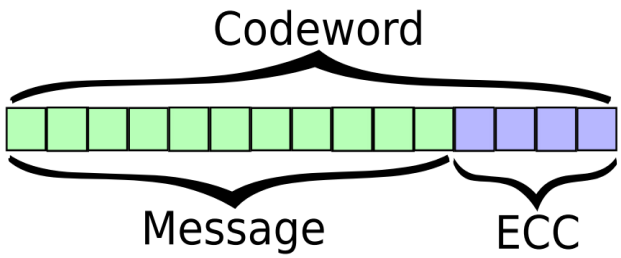

Figure 4.2 BCH codeword structure.

The encoding and decoding BCH codes is performed by utilizing limited fields. A short diagram of finites fields is fundamental in understanding both the component of BCH codes and the proposed enhancement.

I. SYNTHESIS RESULTS

The simulation of the proposed system has done on the Xilinx ISE synthesis tool the summary of device utilization has given below table 5.1 under the heading device utilization summery.

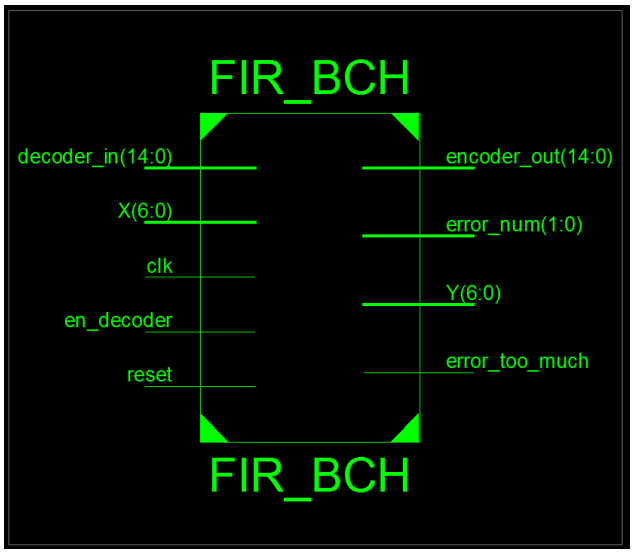

Fig. 4.1 RTL Schematic of Proposed Architecture

Table 4.1 Timing summery of device utilization

```
Device utilization summary:
---------------------------
Selected Device : 4vlx80ff1148-12
 Number of Slices:                      194  out of  35840     0%
 Number of Slice Flip Flops:              9  out of  71680     0%
 Number of 4 input LUTs:                347  out of  71680     0%
 Number of IOs:                          50
 Number of bonded IOBs:                  50  out of    768     6%
    IOB Flip Flops:                      16
 Number of GCLKs:                         1  out of     32     3%
```

Table 5.2: Comparison of Parameters with Existing Work with Eleven FIR Filters

Parameters	Proposed Work	Existing Work
Slices	194	14422
Flip Flops	9	6478
LUTs	347	28331
Frequency	780.762 MHz	-
Delay	14.633 ns	-

II. CONCLUSION AND FUTURE SCOPES

The fault tolerant architecture of the proposed system with the utilization of 15 parallel FIR filter architecture and BCH codes are performed well in terms of area than the previous systems. The proposed architecture has lower complexity in architecture while working faster. The frequency of circuit is 780.762 MHz. The scheme can be used for parallel filters that have the same response and process different input signals. The proposed scheme can also be applied to the IIR filters. Future work will consider the evaluation of the benefits of the proposed technique for IIR filters. The extension of the scheme to parallel filters that have the same input and different impulse responses is also a topic for future work.

REFERENCES

[1] Z. Gao et al., "Fault Tolerant Parallel Filters Based on Error Correction Codes," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 2, pp. 384-387, Feb. 2015.

[2] Z. Gao, W. Yang, X. Chen, M. Zhao, and J. Wang, “Fault missing rate analysis of the arithmetic residue codes based

fault-tolerant FIR filter design," in Proc. IEEE IOLTS, Jun. 2012, pp. 130-133.

[3] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Strutural DMR: A technique for implementation of soft-error-tolerant FIR filters," IEEE Trans. Circuits Syst., Exp. Briefs, vol. 58, no. 8, pp. 512-516, Aug. 2011.

[4] Y.-H. Huang, "High-efficiency soft-error-tolerant digital signal process¬ing using fine-grain subword-detection processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 2, pp. 291-304, Feb. 2010.

[5] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. IEEE IOLTS, Jul. 2008, pp. 192-194.

[6] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 4, pp. 336-348, Apr. 2006.

[7] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405-418, Sep. 2005.

[8] A. Reddy and P. Banarjee "Algorithm-based fault detection for signal processing applications," IEEE Trans. Comput., vol. 39, no. 10, pp. 1304-1308, Oct. 1990.

[9] T. Hitana and A. K. Deb, "Bridging concurrent and non-concurrent error detection in FIR filters," in Proc. Norchip Conf., 2004, pp. 75-78. [9]

[10] P. P. Vaidyanathan. Multirate Systems and Filter Banks. Upper Saddle River, NJ, USA: Prentice-Hall, 1993. [10] A. Sibille, C. Oestges, and A. Zanella, MIMO: From Theory to Imple-mentation. San Francisco, CA, USA: Academic Press, 2010.

[11] P. Reviriego, S. Pontarelli, C. Bleakley, and J. A. Maestro, "Area efficient concurrent error detection and correction for paral¬lel filters," IET Electron. Lett., vol. 48, no. 20, pp. 1258-1260, Sep. 2012.

[12] A. V. Oppenheim and R. W. Schafer, Discrete Time Signal Processing. Upper Saddle River, NJ, USA: Prentice-Hall 1999.

[13] S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall. 2004.

[14] R. W. Hamming, "Error correcting and error detecting codes," Bell Syst. Tech. J., vol. 29, pp. 147-160, Apr. 1950.

[15] R.C. Baumann. Radiation Induced Soft Errors in Advanced Semiconductor Technologies. IEEE Transactions on Device and Materials Reliability, 5(3):305-316, Sept. 2005.