Research Result

A Fault Tolerance PMVM Design via Vedic Multiplication

Minku Kumar[1], Prof. Ashish Raghuwanshi[2]

[1]*M. Tech. Research Scholar, Department of Electronics & Communication Engineering, IES College of Technology, Bhopal*
[2]*Research Guide, Department of Electronics & Communication Engineering, IES College of Technology, Bhopal*

ABSTRACT

To resolve matrix calculations, many systems use common procedures such as matrix processing in parallel. These are often known as parallel matrix-vector multiplications (PMVM), and they are employed in various signal-processing processes and systems that deal with information such as images and data. These processes have the potential to cause or introduce faults in the signal or information, which must be addressed. As a result, PMVM must have an error control system. FPGAs are commonly used to create digital circuitry for this purpose, and these flaws are referred to as faults in the circuits. As a result, the suggested architecture in this work is equipped with an error control and fault tolerance scheme based on Vedic multiplication operations and an optimized area in terms of LUTs and Registers. After synthesis on XILINX, the suggested architecture outperforms earlier designs.

KEYWORDS

PMVM, FPGA, fault tolerance, matrix, Vedic multiplication, Error detection and correct

1. INTRODUCTION

Advances in VLSI technology, like advances in digital mobile and embedded systems, are on the increase. Digital signal processing is found in all complex digital systems. In all of these approaches, faster multiplication is critical. Multiplication is the most basic and often utilised Arithmetic activity in digital devices. It also serves as the foundation for more complex operations such as convolution, DFT, FFT, Walsh Hadamard transformations, and so on. As a result, the engineers are continually working on the implementation of new models, algorithms, and hardware. Using vedic sutras for multiplication is one such solution. Vedic Mathematics is one of the earliest methods of mathematical approach. The Vedic approach to mathematics is absolutely unique and quite similar to how the human mind works.

Field Programmable Gate Arrays (FPGAs) are a stage in the evolution of Integrated Circuits (ICs). FPGAs are reprogrammable silicon chips that may be modified to perform customised hardware functionality of any digital circuit. FPGAs are extremely attractive for digital circuit implementation because of their flexibility, programmability, capacity for numerous applications, and low-end product cycle. The primary distinction between FPGAs and traditional fixed logic solutions such as Application Specific Integrated Circuits (ASICs) is that the FPGA may be programmed on-site. Using an FPGA instead of a fixed logic solution minimizes non-recurring engineering (NRE) expenses and reduces time-to-market dramatically.

The fact that FPGAs integrate the greatest aspects of ASICs and processor-based systems drives their proliferation across all sectors. These reprogrammable silicon chips offer the same flexibility as software operating on a processor-based system, but they are not constrained by the number of available computing cores. The programming environment is provided by software tools, but FPGA circuitry provides a true "hard" implementation of program execution.

The FPGA architecture is made up of several logic modules that are arranged in an array structure, and these modules are reconfigurable on-site, thus the name Configurable Logic Blocks (CLBs). Routers employ the channels between the CLBs. CLB arrays are surrounded by programmable I/O modules and linked via programmable interconnects. FPGA architecture is divided into two subclasses based on the granularity of CLBs: coarse-grained FPGAs and fine-grained FPGAs.

Advances in VLSI technology, like advances in digital mobile and embedded systems, are on the increase. Digital signal processing is found in all complex digital systems. In all of these approaches, faster multiplication is critical. Multiplication is the most basic and often utilised Arithmetic activity in digital devices. It also serves as the foundation for more complex operations such as convolution, DFT, FFT, Walsh Hadamard transformations, and so on. As a result, the engineers are continually working on the implementation of new models, algorithms, and hardware. Using vedic sutras for multiplication is one such solution. Vedic Mathematics is one of the earliest methods of mathematical approach. The Vedic approach to

mathematics is absolutely unique and quite similar to how the human mind works.

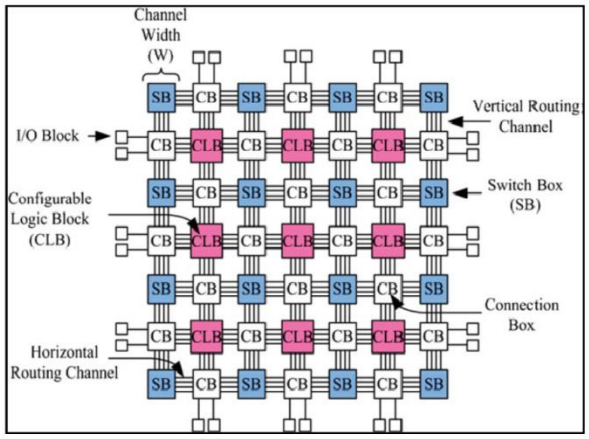

Fig. 1.1 General Architecture of FPGA.

Field Programmable Gate Arrays (FPGAs) are a stage in the evolution of Integrated Circuits (ICs). FPGAs are reprogrammable silicon chips that may be modified to perform customised hardware functionality of any digital circuit. FPGAs are extremely attractive for digital circuit implementation because of their flexibility, programmability, capacity for numerous applications, and low-end product cycle. The primary distinction between FPGAs and traditional fixed logic solutions such as Application Specific Integrated Circuits (ASICs) is that the FPGA may be programmed on-site. Using an FPGA instead of a fixed logic solution minimises non-recurring engineering (NRE) expenses and reduces time-to-market dramatically.

The fact that FPGAs integrate the greatest aspects of ASICs and processor-based systems drives their proliferation across all sectors. These reprogrammable silicon chips offer the same flexibility as software operating on a processor-based system, but they are not constrained by the number of available computing cores. The programming environment is provided by software tools, but FPGA circuitry provides a true "hard" implementation of program execution.

The FPGA architecture is made up of several logic modules that are arranged in an array structure, and these modules are reconfigurable on-site, thus the name Configurable Logic Blocks (CLBs). Routers employ the channels between the CLBs. CLB arrays are surrounded by programmable I/O modules and linked via programmable interconnects. FPGA architecture is divided into two subclasses based on the granularity of CLBs: coarse-grained FPGAs and fine-grained FPGAs.

2. SYSTEM MODEL

Data mining, image processing, scientific computation, and other research domains need matrix multiplication in either dense or sparse form. SpMV is a kind of matrix vector multiplication in which the matrix elements are sparse. Because of the multiplication of zero components, normal MVM implementation takes longer. MVM is utilised in a variety of applications, including binary search trees, graph analysis, network analysis, and, more recently, image processing. MVM performance is mostly determined by the storage format and the efficient use of underlying technology. Parallel computing is supported by all contemporary processor hardware. Two additional techniques for rapid processing and fault tolerance are interfaced with the MVM algorithm to boost performance.

- *Vedic Multiplication*

In Indian culture, old Vedic mathematics is immensely popular. It has several applications in various fields of education such as mathematics, engineering, and so on. Vedic mathematics is a condensed version of the four Vedas. Vedic mathematics is an ancient kind of mathematics that developed in India. It is mostly used to answer mathematical problems in simple methods, resulting in speedier calculations.

Not only is Vedic mathematics a mathematical marvel, but it is also logical. It explains several mathematical terminologies, such as arithmetic, and geometry (plane, coordinate, trigonometry, factorization of quadratic equations, and calculus). As a result, no one can deny it. Because of these distinguishing characteristics, this topic is regarded as a fascinating study field in India and the majority of other nations. Manual computation is sufficient to answer all complicated mathematical problems utilising this shortcut approach, which is more powerful and simple to utilise.

The Vedic Algorithm

Step 1: Make two rows of the number.

Step 2: Next, multiply the rightmost digits of the provided integers vertically and record the product as the answer of the rightmost digit.

Step 3: For both numbers, take two digits from the rightmost digit and cross-multiply the rightmost digit of the multiplicand with the second digit of the multiplier and the second digit of the multiplicand with the rightmost digit of the multiplier, then add the cross-product of the above multiplication.

Step 4: Repeat step 3 with the following digit from the least significant digit until the most significant digit is reached.

Step 5: Finally, multiply the most significant digits of both numbers vertically and record the result as a product of the answer's leftmost digit.

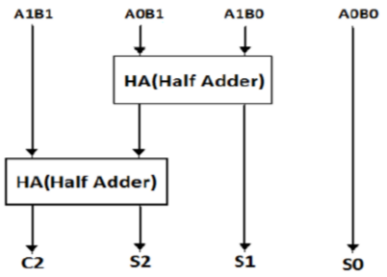

Fig. 2.1 2×2 Binary Multiplication.

- *Fault Tolerance*

Fault tolerance is a characteristic that is built into a system to help it accomplish a certain design goal. A design must not only achieve various functional and performance goals, but it must also meet several other design standards. The

most important extra needs are dependability, availability, safety, formability, maintainability, and testability; fault tolerance is a system-wide property capable of meeting such requirements. Fault tolerance in systems is achieved via the use of two techniques: design variety and redundancy.

A fault-tolerant system is one that can continue to fulfil its given functions even when there are hardware and/or software faults. Fault tolerance is the property that allows a system to operate fault-tolerantly. Finally, the phrase fault-tolerant computing refers to the process of conducting fault-tolerant computations, such as those done by a computer. Error processing and error treatment are used to achieve fault tolerance. Error processing aims to remove mistakes from the computational state before failure occurs; fault treatment aims to prevent faults from being triggered again.

3. PROPOSED METHODOLOGY

To overcome area constraints and errors, an effective design of integer parallel MVM was introduced in the Xilinx ISE design suite in this investigation. The Virtex family has decided to implement the proposed concept in the Virtex 7 hardware. RTL design schematic is recommended. Figure 3.1 shows the top module of Xilinx ISE. The following are the major design constraints to be considered while creating reversible logic doors for vedic multipliers. Reversible Logic gates should cost at least a quantity.

- The design can be optimized to generate minimum trash outputs.
- The logic gates reversible must use a minimum amount of continuous inputs.

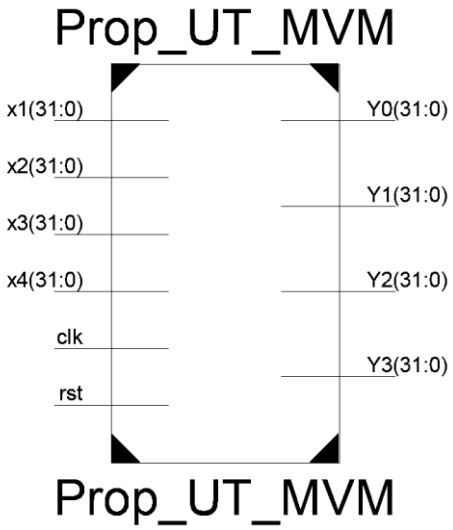

Fig. 3.1 Schematic of Top Module Inputs and Outputs

In proposed the top module x1, x2, x3, x4 is 32-bit input and 32-bit output is y1, y2, y3, y4. Clock is represented by clk and queue in RTL schematic is represented by rst. In Figure 3.2, Figure m1, m2, m3, m4, the schematic of the suggested system architecture shows the FFTs used together with edc (error detection and correction).

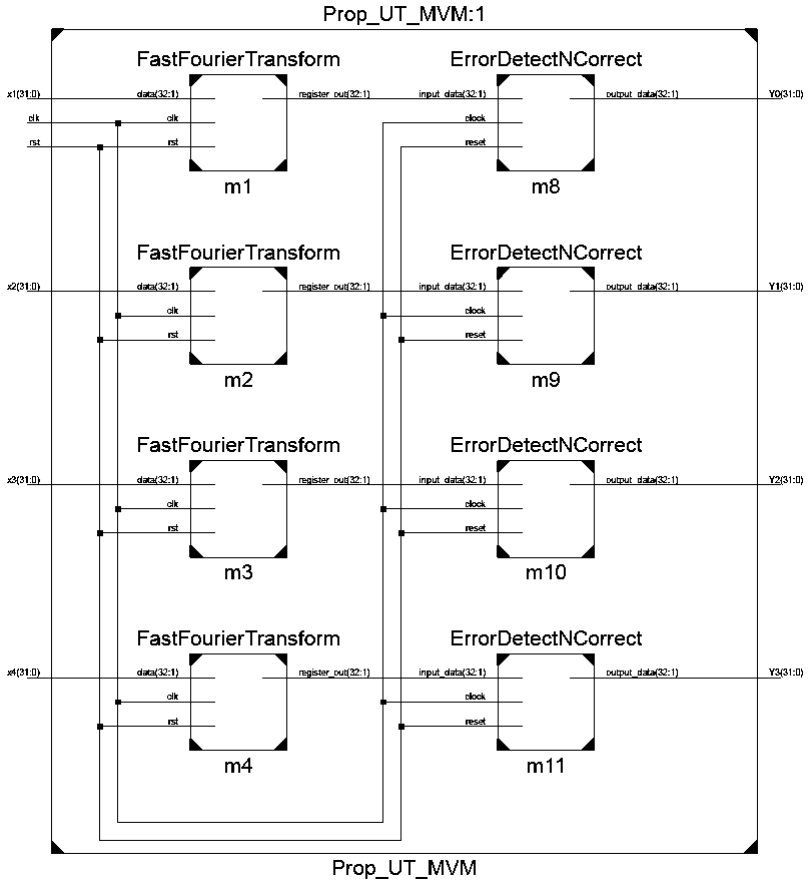

Fig. 3.2 Schematic of Proposed System Architecture

Each FFT has received 32 bit data, and the individual result of each FFT is stored in a 32 bit register. In the proposed work, FFT is implemented using a butterfly method, as illustrated in Fig. 3.3. To implement EDE error correction codes, the internal schematic of the EDC (Error Detect and Correct) Sub Module is depicted in Fig 3.4. The encoder module employs two primary approaches: error correcting code and a multiplexer. In terms of area, the proposed design has implemented architecture is highly efficient. The suggested module's synthesis validates the efficiency of the proposed module in Xilinx.

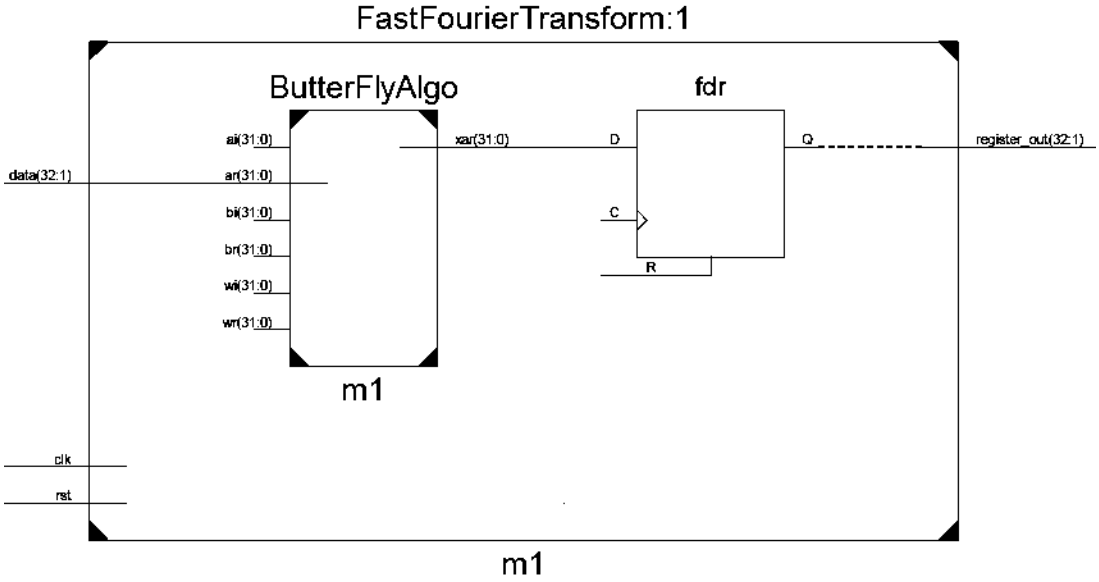

Fig.3.3 Schematic of Fast Fourier Transform (FFT) Module

It theoretically means that input is a spectrum, so it creates an illusion that each channel is modulated by a carrier. At the receiving end the FFT is used to retrieve the original signal. The Radix 2 FFT architecture is flexible to be implemented using SDF, SDC, MDF, MDC architectures and hence it becomes an advantage when it comes to reducing area, delay and power. Reduction in number of multiplications and computations are done by two schemes in the FFT algorithms.

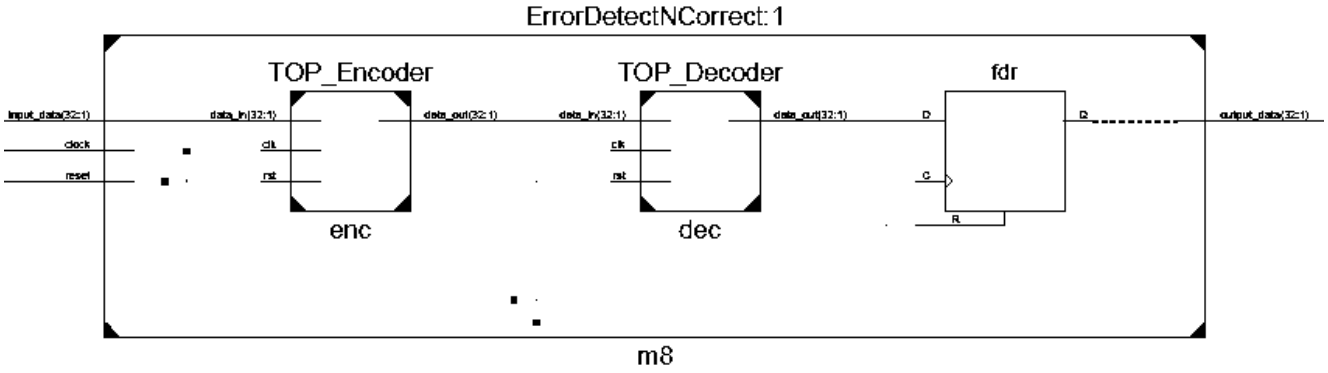

Fig.3.4 Internal Schematic of EDC (Error Detect and Correct) Module

4. SYNTHESIS OUTCOMES

The suggested model was synthesised in Xilinx 13.1 ISE utilising the Virtex device family, with Virtex 7 being employed to do so. Figure 4.1 depicts the suggested design's user interface and device utilisation summary. The suggested design's performance has been analysed using a device utilisation summary. Figure 4.1 depicts the synthesis report of the proposed module. The suggested model's performance was validated by counting the number of registers and LUTs on a Virtex-7 device. To evaluate the performance, a comparison study is performed with earlier work, which reveals that the suggested model has better device utilisation than past work.

With the distributed data structure described above, each CPU executes the following operations to achieve concurrent matrix-vector multiplication. First, according to the data structure and communication strategy provided, each processor runs non-blocking send routines to send the local interface points, x band.

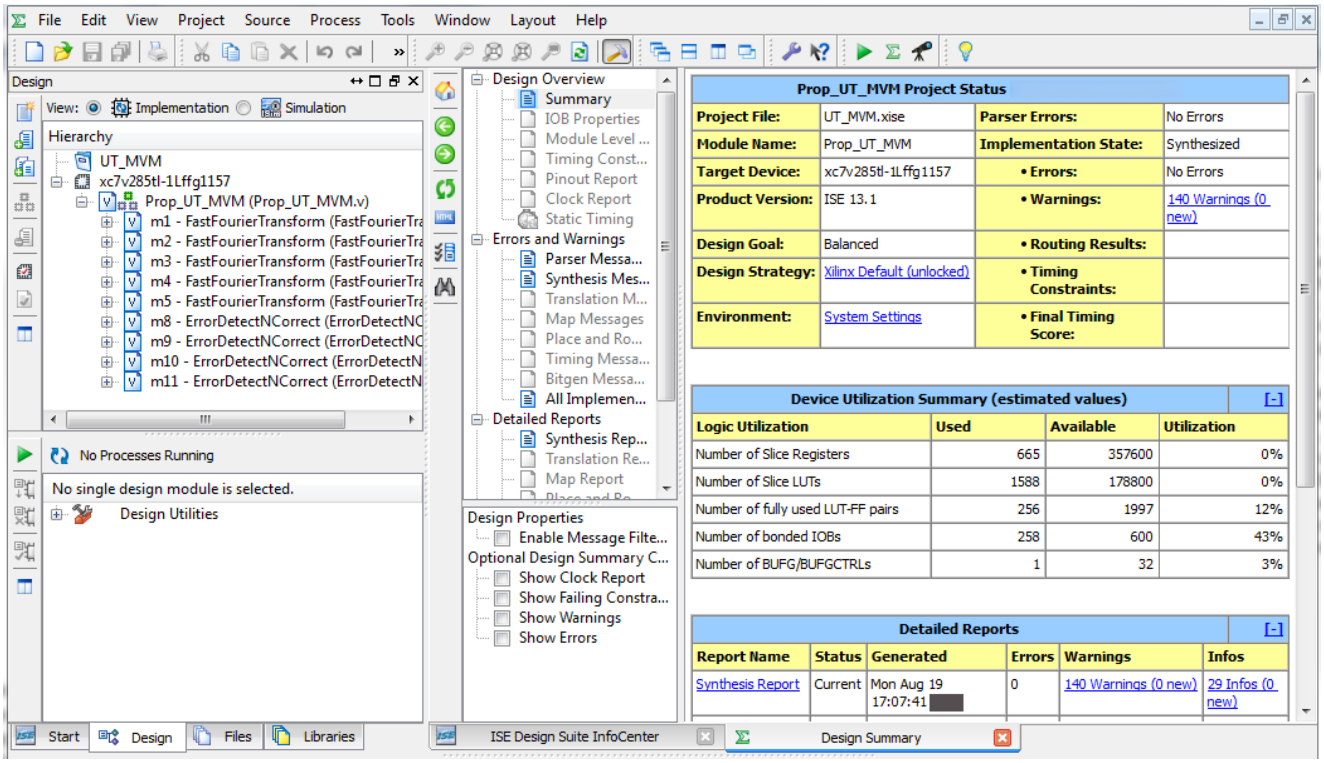

Fig. 4.1 User Interface and Device Utilization Summary of Proposed Design

The comparison tabulated in Table 1 comparison of cost/resources (device utilization) for four parallel MVMs. Proposed model has used 92% less LUTs resources and 87% fewer registers with respect to previous base work.

The graphical analysis of table 1 has shown in Fig. 4.2 bar Graph Cost/Resources (Device Utilization) Comparison.

Table 1: Comparison of Cost/Resources (Device Utilization) For Four Parallel MVMs

Parameters	*Base PaperWork*	*Proposed Work*
Look Up Tables	22024	1588
Registers	5489	665

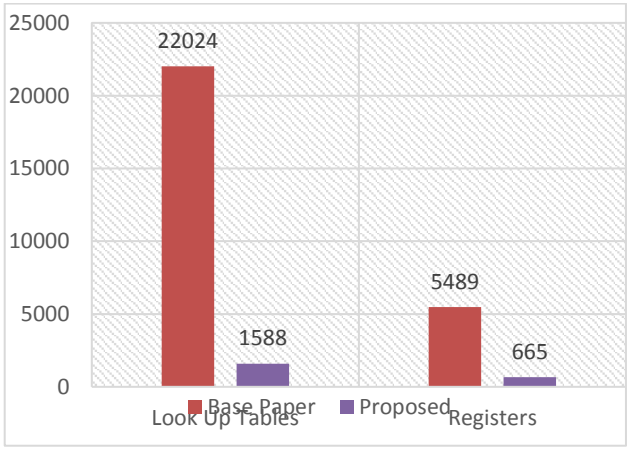

Fig. 4.2 Bar Graph Cost/Resources (Device Utilization) Comparison

5. CONCLUSION AND FUTURE SCOPES

In this paper, we implement an efficient fault tolerance design of integer parallel MVM in the Xilinx 13.1 ISE design suite. The suggested module's software was synthesised using the Virtex-7 device family from Xilinx FPGA. To boost the performance of efficient area MVM applications in terms of LUTs and agglomerated registers, as well as fault tolerance and parallel computing. It also aims to improve the algorithm's efficiency through proposed approaches.

The effectiveness of MVM systems is investigated using vedic multiplication simulation. The MVM approach was utilised to get the best switching angles with the least amount of fault in the device output. The simulation was run, and the results were shown and reviewed. To confirm the simulation matrix's results, a Xilinx-based FPGA model is introduced. As a result, it is discovered that an expanded fault tolerance device is essential for all industrial applications.

REFERENCES

[1] Z. Gao, Q. Jing, Y. Li, P. Reviriego and J. A. Maestro, "An Efficient Fault-Tolerance Design for Integer Parallel Matrix–Vector Multiplications," in IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 26, no. 1, pp. 211-215, Jan. 2018.

[2] I. Sayahi, M. Machhout and R. Tourki, "FPGA implementation of matrix-vector multiplication using Xilinx System Generator," 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET), Hammamet, 2018, pp. 290-295.

[3] S. M. Ali, W. Shaojun, M. Ning and P. Yu, "A bandwidth in-sensitive low stall sparse matrix vector multiplication architecture on reconfigurable FPGA platform," 2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Yangzhou, 2017, pp. 171-176.

[4] Z. Gao, P. Reviriego and J. A. Maestro, "Efficient fault tolerant parallel matrix-vector multiplications," 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), Sant Feliu de Guixols, 2016, pp. 25-26.

[5] A. Schöll, C. Braun, M. A. Kochte and H. Wunderlich, "Efficient Algorithm-Based Fault Tolerance for Sparse Matrix Operations," 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, 2016, pp. 251-262.

[6] C. Yang, Y. Wang and J. D. Owens, "Fast Sparse Matrix and Sparse Vector Multiplication Algorithm on the GPU," 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, Hyderabad, 2015, pp. 841-847.

[7] P. N. Q. Anh, R. Fan and Y. Wen, "Reducing Vector I/O for Faster GPU Sparse Matrix-Vector Multiplication," 2015 IEEE International Parallel and Distributed Processing Symposium, Hyderabad, 2015, pp. 1043-1052.

[8] J. Huang, J. Ren, W. Yin and L. Wang, "No zero padded sparse matrix-vector multiplication on FPGAs," 2014 International Conference on Field-Programmable Technology (FPT), Shanghai, 2014, pp. 290-291.

[9] Z. Gao et al., "Fault tolerant parallel filters based on error correction codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 2, pp. 384–387, Feb. 2015.

[10] Z. Gao et al., "Fault tolerant parallel FFTs using error correction codes and Parseval checks," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 2, pp. 769–773, Feb. 2016.

[11] Z. Gao, P. Reviriego, and J. A. Maestro, "Efficient fault tolerant parallel matrix-vector multiplications," in Proc. IEEE 22nd Int. Symp. On-Line Test. Robust Syst. Design (IOLTS), Jul. 2016, pp. 25–26.