Parallel Matrix Vector Multiplication Fault Tolerant Design

Kumar Avinash[1], Prof. Swati Gupta[2]

[1]*M.Tech Scholar,* [2]*Research Guide*

Department of Electronics and Communication Engg. Vidhyapeeth Institute of Science & Technology, Bhopal

Abstract – Many system are using typical operations like matrix processing in parallel to resolve the matrix calculations. These are usually named as parallel matrix vector multiplications (PMVM), and it is used in many signal processing operations and systems which belongs to processing of information like image and data. These operations can generate or introduce errors in the signal or information which needs to be corrected. So PMVM must be equipped with the error control mechanism within. To make digital hardware for this FPGAs are generally preferred and these errors named as faults in the circuits. So in this work proposed design is equipped with the error control and fault tolerance scheme using Vedic multiplication operations with the optimized area in terms of LUTs and Registers.After synthesis on XILINX proposed architecture shows the merits over previous designs.

Keywords –PMVM, FPGA, fault tolerance, matrix, Vedic multiplication, Error detect and correct.

I. INTRODUCTION

As with the Advances in VLSI technology are on the rise for digital mobile and embedded systems. In all the complicated digital systems, digital signal processing is present. Faster multiplication in all these schemes is very essential. Multiplication is the basic and most Arithmetic activities in digital devices are commonly used. It is also the foundation for complicated activities such as convolution, DFT, FFT, hadamard transformations from Walsh, etc. The developers are therefore constantly working for the implementation of fresh models, algorithms and hardware. One such solution is to Use vedic sutras for multiplication activities. Vedic Mathematics is one of the oldest mathematical methodology techniques used. The Vedic approach to mathematics is completely distinct and very close to how a human mind operates..

Field Programmable Gate Arrays (FPGAs) are a step in the continuum ofevolution of Integrated Circuits (IC). FPGAs are reprogrammable silicon chips and are one of the Programmable Logic Devices (PLDs) that can be configured to implement customized hardware functionality of any digital circuit. Due to their flexibility, programmability, capacity for various applications and low end product cycle, FPGAs are highly desirable for implementation of digital circuits. The main difference between FPGAs and conventional fixed logic implementations, such as Application Specific Integrated Circuits (ASICs), is that the designer can program the FPGA on-site. Using an FPGA instead of a fixed logic implementation eliminates the non-recurring engineering (NRE) costs and significantly reduces time-to-market.

FPGA chips adoption across all industries is driven by the fact that FPGAs combine the best parts of ASICs and processor-based systems. These reprogrammable silicon chips also have the same flexibility of software running on a processor-based system, but it is not limited by the number of processing cores available. The software tools provide the programming environment, whereas FPGA circuitry is truly a “hard” implementation of program execution.

The FPGA-architecture consists of many logic modules which are placed in array- structure and these modules are configurable at site and are therefore called as Configurable Logic Blocks (CLBs). The channels between the CLBs are used for routing. The arrays of the CLBs are surrounded by programmable I/O modules and connected via programmable interconnects. There are two subclasses of FPGA architecture depending on granularity of CLBs: Coarse-grained and Fine-grained FPGAs.

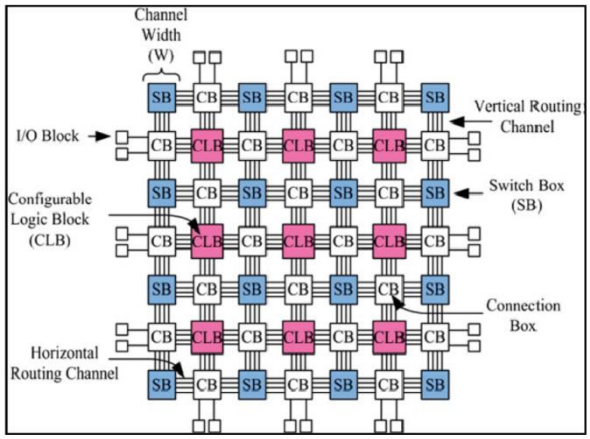

Fig. 1.1 General Architecture of FPGA.

The coarse-grained FPGAs have very large logic modules/CLBs with sometimes two or more sequential logic elements, whereas the fine-grained FPGAs have very

simple logic modules. A conventional, island-style FPGA can be viewed as an array of CLBs connected by programmable interconnects i.e. switchboxes (SBox) and connection boxes as shown in Figure 1.1 .As shown in Figure 1.1, N programmable lookup tables (LUTs) are connected together using internal interconnect inside the CLB. The LUT is simply a circuit that selects the output of aStatic Random Access Memory (SRAM) cell based on the LUT's k inputs: by programming appropriate values in the SRAM cells the LUT can implement any k-input function.

Parallel matrix-vector multiplication is an important operation because it is involved in many scientific computations. Some examples are the solution of linear system with iterative methods, network applications and linear programming. Parallel matrix-vector multiplication software is available on single processor systems, however, yet there is no standard. There are several reasons. First, the matrix-vector multiplication is often not a very complex operation. That is, users can frequently write their own routine. Another reason is that different users and different parallel matrix storage formats convenient for their application and there are no widely agreed upon storage formats. On parallel computer systems very few general parallel matrix-vector packages are available. They are often mainly architecture dependent and depend strongly on data structure. As a result they can hardly be useful for a general case. In this work we propose a parallel matrix-vector multiplication algorithm based upon the standard Message Passing Interface MPI.

II. SYSTEM MODEL

Research fields such as data mining, image processing, scientific computation etc. require matrix multiplication either in dense or sparse form. SpMV is a normal matrix vector multiplication in which the elements of the matrix are sparse. Normal implementation of MVM consumes more time due to the multiplication of zero elements. MVM is used in applications such as binary search tree, graph analysis, network analysis and recently in image processing also. Performance improvement of the MVM is mainly dependent on the storage format and efficient utilization of underlying hardware. All current processor hardware has the parallel computing capability. To improve the performance two addition approaches are interfaced with MVM algorithm for fast processing and fault tolerance.

- *Vedic Multiplication*

The ancient Vedic mathematics is very popular in Indian culture. It has many applications in different branches of education like mathematics, engineering, etc. The Vedic mathematics is a summary of four Vedas. This Vedic mathematics is one of the ancient forms of mathematics which originated in India. It is mainly used to solve the mathematical problems in simple ways, thus it leads to get the result of faster computation.

Vedic mathematics is not only a mathematical wonder, but also it is logical. It gives explanation about several mathematical terms, including arithmetic, geometry (plane, co-ordinate, trigonometry, factorization of quadratic equations and even calculus. That‘s why it cannot be disproved by anyone. Due to these unique qualities, this field is treated as a very interesting research field in India and most of the foreign countries. By using this shortcut technique, manual calculation is enough to solve all the complex mathematical problems in the easiest way and it is more powerful and simple to use.

Vedic Algorithm

Step 1: Write the number in two rows.

Step 2: The next step is to multiply the rightmost digits of the given numbers vertically, and note down the product as the answer of rightmost digit.

Step 3: Consider two digits from the rightmost digit for both the numbers 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 and add the cross product of the above said multiplication.

Step 4: Consider the next digit from the least significant digit and do the step 3 and repeat step 3 until the most significant digit is reached.

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

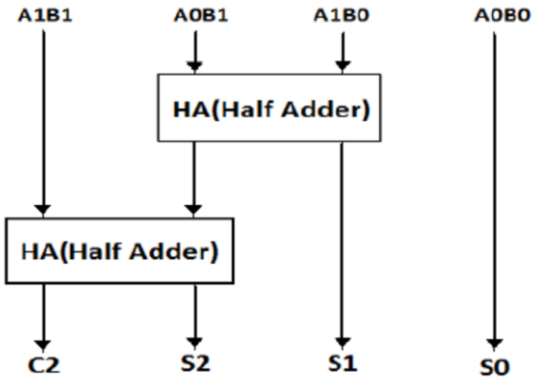

Fig. 2.1 2×2 Binary Multiplication.

- *Fault Tolerance*

Fault tolerance is the attribute that is designed into a system to achieve a certain design goal. Just as a design must meet many functional and performance goals, it must also satisfy numerous other design requirements. The most

prominent of the additional requirements are reliability, availability, safety, per formability, maintainability and testability; fault tolerance is a one- system- attribute capable of fulfilling such requirements. Fault tolerance is achieved in systems using two techniques namely design diversity and redundancy.

A fault-tolerant system is one that can continue the correct performance of its specified tasks even in the presence of hardware and/or software failures. Fault tolerance is the attribute that enables a system to achieve fault-tolerant operation. Finally, the term fault-tolerant computing is used to describe the process of performing calculations, such as those performed by a computer, in a fault-tolerant manner. Fault tolerance is carried out by error processing and error treatment. Error processing is aimed at removing errors from the computational state, if possible before failure occurrence; fault treatment is aimed at preventing faults fr**o**m being activated again.

III. PROPOSED METHODOLOGY

In this analysis, an effective design of integer parallel MVM has been introduced in Xilinx ISE design suite to overcome area restriction and faults. The Virtex family has selected to introduce the suggested design in the Virtex 7 hardware. RTL design schematic suggested Top module was shown in Fig in Xilinx ISE. 3.1.

The Following are the significant design limitations to be regarded when developing 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.

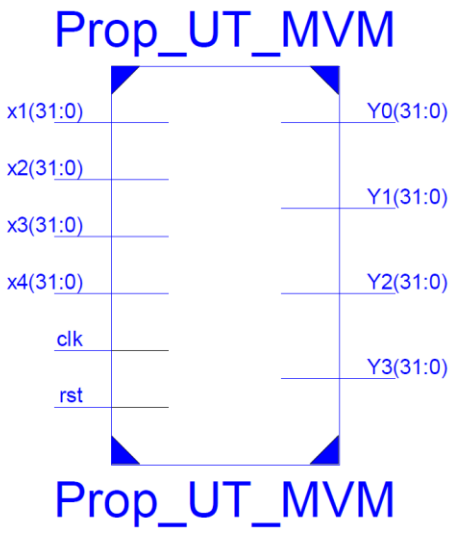

Fig. 3.1 Schematic of Top Module Inputs and Outputs

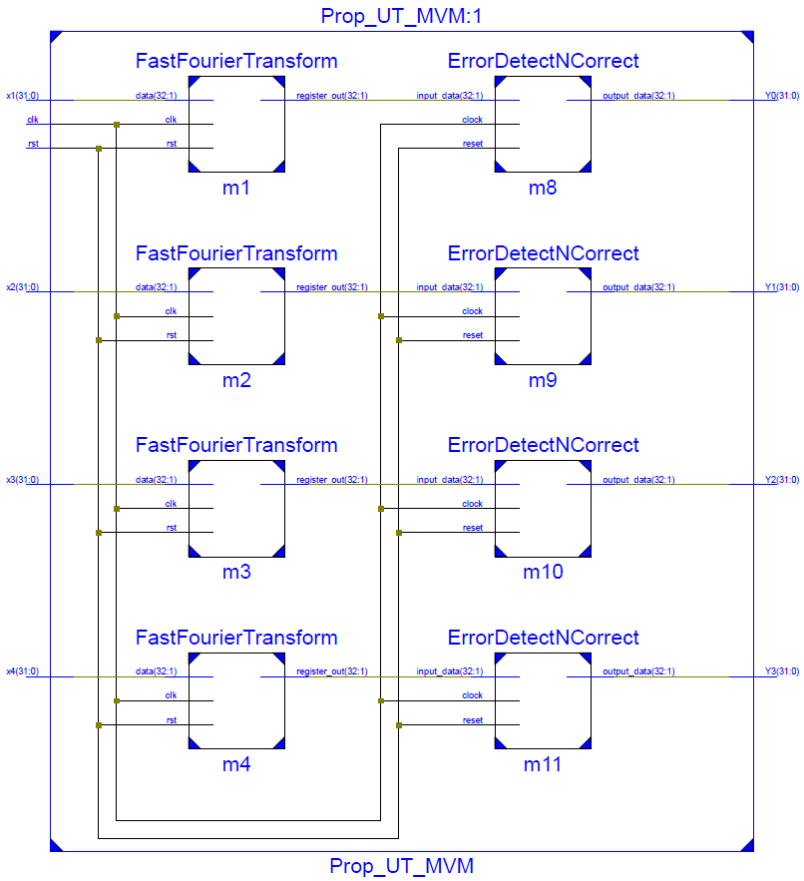

Fig. 3.2Schematic of Proposed System Architecture

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).

For each FFT 32 bit data has taken and individual output of each FFTs are stored in 32 bit register. Sub module of FFT module has shown in Fig. 3.3, FFT is implemented using a butterfly algorithm in proposed work. Internal Schematic of EDC (Error Detect and Correct) Sub Module has shown in Fig 3.4. to implement EDE error correction codes are used. In encoder module there are two fundamental approaches are used which are error correction code and a multiplexer. The proposed has design has implemented architecture is very efficient in terms of area. Synthesis of proposed module confirms the efficiency of the proposed module in Xilinx.

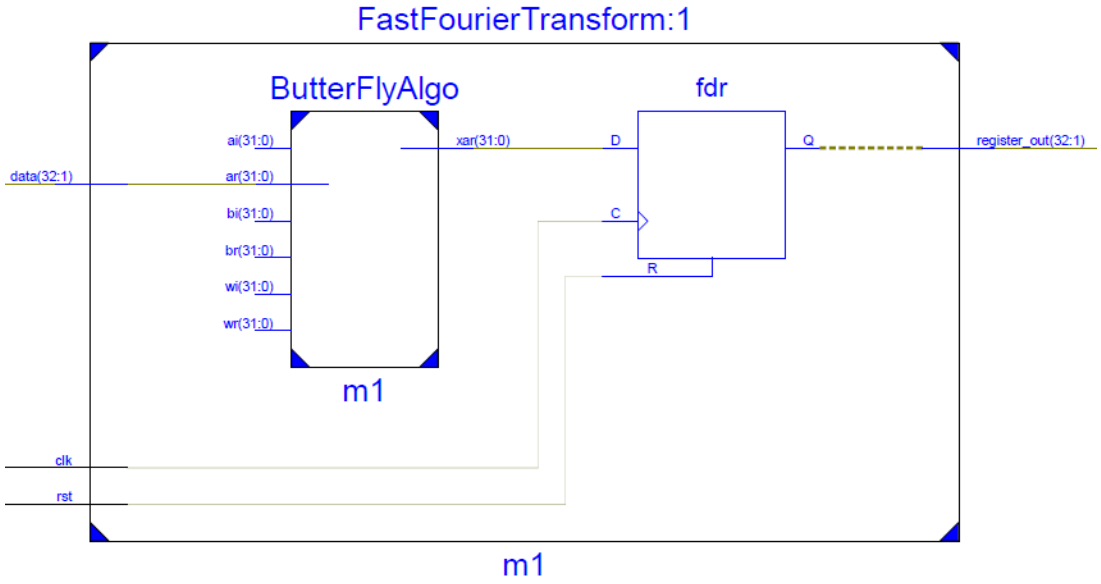

Fig.3.3 Schematic of Fast Fourier Transform (FFT)Module

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.

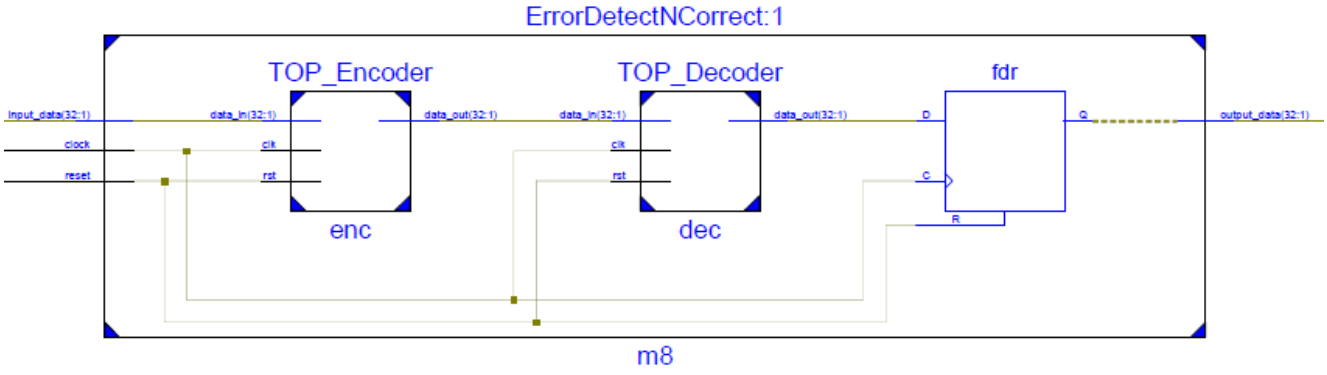

Fig.3.4 Internal Schematic of EDC (Error Detect and Correct) Module

IV. SYNTHESIS OUTCOMES

Synthesis of proposed model has done in Xilinx 13.1 ISE using Virtex device family to synthesize proposed design Virtex 7 has used. Fig. 4.1 shows the user interface and device utilization summary of proposed design. The performance of proposed design has been evaluated based on device utilization summary. In figure 4.1 Synthesis report of proposed module has shown. The performance of proposed model has verified based on number of register and LUTs counts using Virtex-7 device. In order to examine the performance a comparative analysis is carried out with previous work which confirms that proposed model has better device utilization as compared to previous work.

To perform the parallel matrix-vector multiplication, with the distributed data structure described above, each processor performs the following operations. First, each processor executes non-blocking send routines to send the local interface points, x band, according to the data structure and communication scheme described.

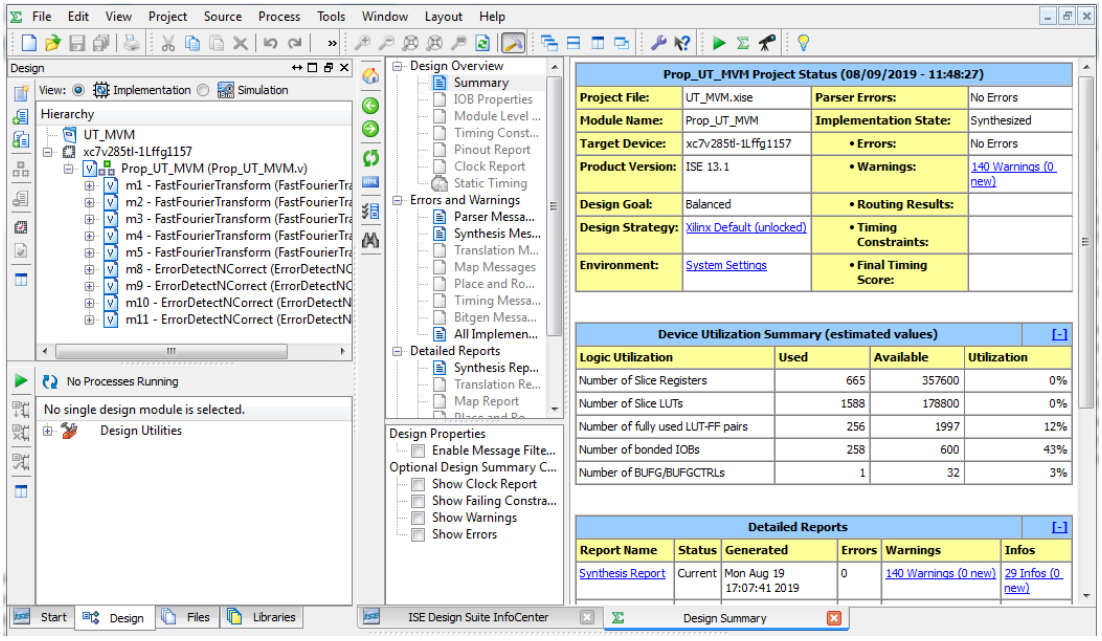

Fig.4.1User Interface and Device Utilization Summary of Proposed Design

The comparison has tabulated in Table 1 comparison of cost/resources (device utilization) for four parallel MVMs. Proposed model has used 92% less LUTs resources and 87% less 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 Paper Work***	***Proposed Work***
Look Up Tables	22024	1588
Registers	5489	665

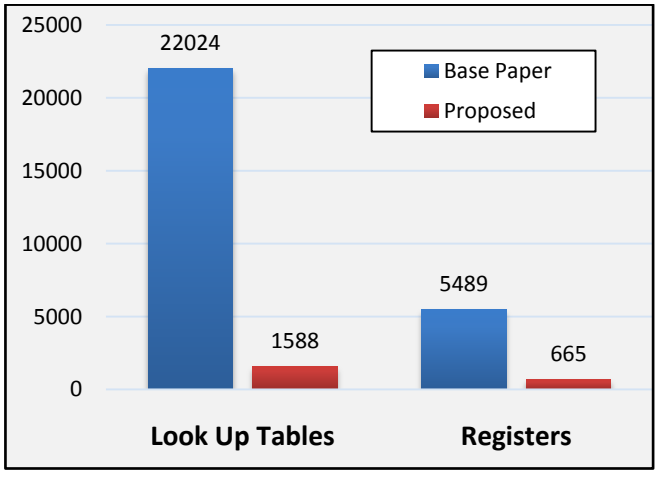

Fig. 4.2Bar Graph Cost/Resources (Device Utilization)Comparison

V. CONCLUSION AND FUTURE SCOPES

In this work Implementation in Xilinx 13.1 ISE design suite of an efficient fault tolerance design of integer parallel MVM. The software synthesis of the proposed module was carried out in Xilinx FPGA's Virtex-7 device family. To improve the performance of efficient area MVM applications in terms of LUTs and agglomerated registers with fault tolerance and parallel computing. It also seeks to enhance the algorithm's efficiency by means of suggested methods.

The MVM efficiency systems are researched using vedic multiplication simulation. The MVM method was used to achieve the optimum switching angles for minimal fualts in the device output.The simulation was conducted and the outcomes are displayed and evaluated. Xilinx based on FPGA model is introduced to validate the outcomes of the simulation matrix. It is therefore found that an enhanced fault tolerance device is prominent 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.