International Journal of Electrical Engineering and Ethics- Volume 8 Issue 6, November - December - 2025

LOW COMPLEXITY ERROR DETECTION IN SYSTOLIC ARRAY ARCHITECTURE FOR MATRIX MULTIPLICATION

JAYABHARATHI
jayabharathi9688@gmail.com

Abstract— **The various engineering applications demanding fault tolerance and the high performance which is given by the matrix multiplication MxM. There will be a significant overhead in conventional method foe a reliable MxM. Especially in FPGAs the improvements are made in the conventional algorithm in hardening solutions but there may be an optimal persistent fault. The proposed work discusses error detection technique of MxM. Compared to prior algorithm-based technique the proposed system leverages the fault model analysis which reduces both algorithmic and architectural costs. Furthermore, at application levels the proposed technique gives the minimal arithmetic overhead and improved efficiency and the error correction is high .**

Keywords: MxM, efficiency, FPGA, Convolutional Neural Networks, error correction, algorithm, fault model, high performance, fault tolerance.

I. INTRODUCTION

Various critical applications of real time safety including computer vision and autonomous system that underpinned by the Matrix Multiplication MxM. To ensure reliable operation of this application which not only demand for the high performance but also fault tolerance. Conventional method achieves fault tolerance often with the incur significant overhead. The proposed work reliable on MxM which particularly focuses on the FPGAs because of their unique performance, re-program-ability and power efficiency. The faults raised from radiation exposure in FPGA are addressed in this paper. It requires innovative solutions for the error detection and disrupt communications. A multi-level fault propagation model that are analyzed that a fault which impact the different layers of computing layers are also discussed in this section. This technique is specially designed for the systolic array on SRAM-based FPGAs with the low overhead “light” Algorithm-Based Fault Tolerance (ABFT) solution. To facilitate the use of reliable “MxM” cores the open-source systolic array design implementation is presented in this paper along with the supporting tools.

The paper structure is outlined in detail as follows. Brief surveys of the various existing methods are discussed in Section II. Section III delves the proposed design methodology. Section IV describes the evaluation of results and analysis. Section V discusses the conclusions of the study.

II. RELATED WORKS

CHARACTERIZATION AND MITIGATION OF SINGLE EVENT TRANSISTORS IN SRAM-BASED FPGA

The impact of single event transistors on SRAM based FPGAs in context with space applications are investigated in this paper. During testing continuous monitoring and control is done with SET characterization and mitigation. For design verification and online reconfiguration this approach ensures proper device under test. The convolutional research focuses mainly on three radiation effects. They are, Single event upsets (SEUs), latch-up and total ionizing dose effects.

Within the realistic radiation environment SET probability is offered by the quantitative assessment. At low linear energy transistors Both SET and SEU are analyzed in cross section. Alpha source exposure and Heavy ion irradiation at the UCL Cyclotron are utilized in these experiments. Due to dominant SEU rate low-LET SET examinations on SRAM is essential. Various filters are used to mitigate SETs which reduces the transient effect. For satellite applications these experiments can aid a radiation immune FPGA platform [1].

SYSTOLIC ARRAY DNN ACCELERATORS- ANALYZING AND MITIGATION OF PERMANENT FAULTS

Though the DNN-Deep Neural Networks are gaining a significant traction because of their effectiveness their computational necessities the hardware acceleration. Systolic array-based matrix multiplication units are equipes the Tensor Processing Unit (TPUs) involved in the deep neural networks.

Particularly for the technologies which are prone to defects are achieving a fault tolerance . Even with the minimal fault rates the research gives the sustainable decrease in the classification accuracy. Without impacting real time performance this approach enables TPU in significantly higher fault rates by maintaining a classification accuracy [2].

FRAME-TO-FRAME CORRELATION WITH ERROR DETECTION IN CNN

In autonomous vehicle Convolutional Neural Networks (CNNs) are vital for object detection. Due to high computational demands real time detection with CNN leaves the limited space for the complex error detection mechanisms.

This approach correlates between consecutive frames and CNNs output in error free scenarios. The object detection is very similar if the difference between two subsequent image is minimal. Error is indicated by the deviation in the correlation. The threshold for the inner frame and output correlation is established and evaluated in this research paper. Using the data from the past radiation experiments detection is done. With the minimal overhead the image processing hardware achieves the 80% error detection [3].

CONVOLUTIONAL NEURAL NETWORKS- ANALYZING AND INCREASING THE RELIABILITY

Safety critical application uses the convolutional neural networks that makes an increased prevalence of Graphics Processing Unit (GPU) for image detection increases focus on reliability. Object detection algorithms running on NVIDIA GPU and improved strategies are evaluates the reliability.

You Only Look Once (YOLO), Faster Region based CNN and residual networks are the object detection algorithms are evaluated. To understand the fault propagation within CNNs and competing the finding with fault injection techniques and the reliability is assessed by exposing live hardware to neutron beams [4].

ALGORITHMIC ERROR DETECTIO-MAKING CONVOLUTIONS MORE ROBUST

The hardware errors become crucial as the convolutional neural networks are used in the safety critical systems to ensure their resilience. The cost is hefty as it replicates the entire CNN guarantees accuracy in computation.

The most computationally expensive techniques use the algorithm-based error detection (ABED) technique is used specifically used for the convolutions. This paper delves about the various ABED variations with each offers a balance between implementation, error coverage and runtime overhead. The output corrupted by the transient hardware errors that detects the ABED effectively [5].

SRAM BASED ERROR DETECTION- ENHANCING ERROR DETECTION IN FFT

The functionality of SRAM based FPGAs are disrupted by the external factors that cause soft errors. Dual modular redundancy are the conventional error detection resource intensive methods.

Fast Fourier Transforms (FFT) algorithm executed on the SRAM based FPGAs which uses a scheme for implementation of the Concurrent Error Detection (CED). To identify errors the CED techniques has the inherent mathematical relationships between the inputs and outputs of FFT algorithm. Based on the mathematical properties of the FFT these CED techniques involve checks which uses the error detection without using the extensive resources [6].

III. METHODOLOGY

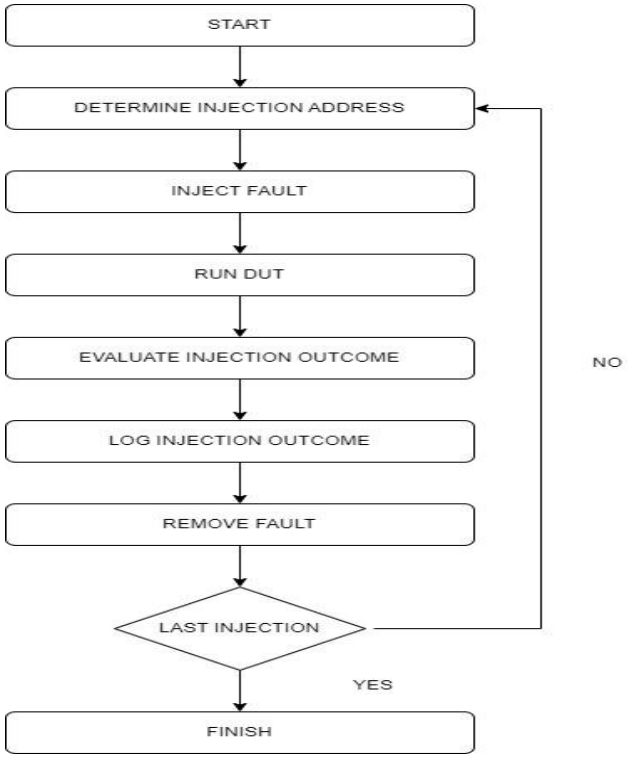


Fig 3.1 Flowchart of Fault injection of FPGA

Fig 3.1 shows the flowchart of fault injection of FPGA. It contains injection address determination, inject fault, Run DUT, evaluate injection outcome, log injection outcome, remove fault and last injection.

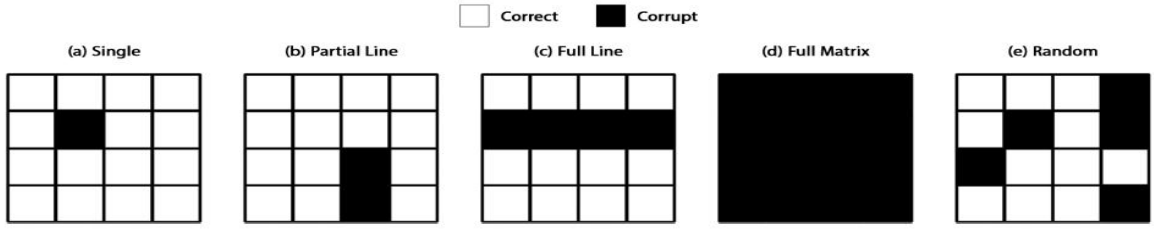


Fig 3.2 Matrix Multiplication output categories

Fig 3.3 shows the matrix multiplication output categories after fault injection. This includes, Single event corruption, full line corruption, Partial row/column corruption, full matrix corruption and random corruption. Full line shows that the fully corrupted one row/column. Full matrix means that the whole matrix is corrupted and the random shows that none of them are corrupted.

TABLE I. Error categories distribution with architectural faults

Single	Partial line	Full line	Full Matrix	Random
27.07%	43.28%	26.687%	0.25%	2.72%

Table I shows the error categories distribution with architectural cost in single, partial, full line, full matrix and random corruption.

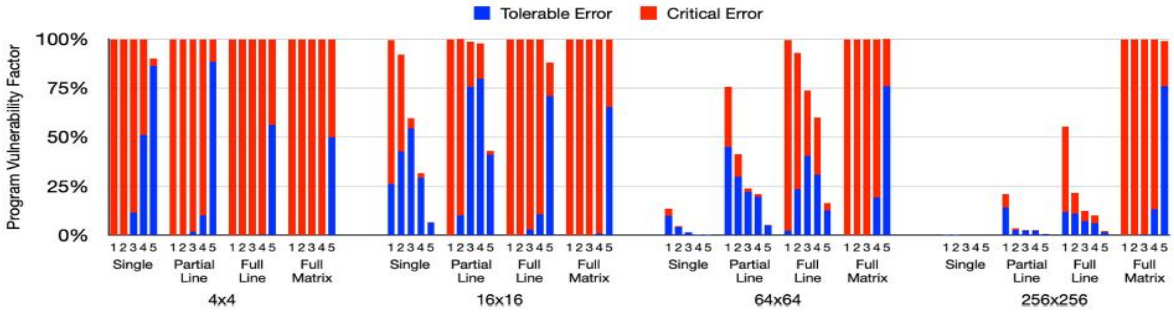


Fig 3.3 LeNet CNN case study

Fig 3.3 LeNet with CNN case study which shows program vulnerability factors and tolerance and critical error of the LeNet with CNN.

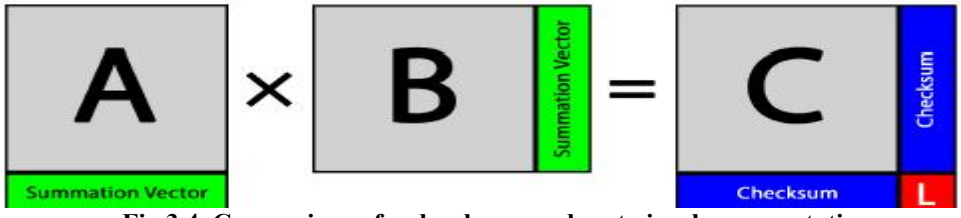


Fig 3.4. Comparison of redundancy and cost visual representation

Fig 3.4 shows the comparison of redundancy and cost visually with it calculates only L instead of the entire checksum.

$C_1 = X_2 \oplus X_3 \oplus X_4$ ----------(1)

$C_2 = X_1 \oplus X_3 \oplus X_4$ ----------(2)

$C_3 = X_1 \oplus X_2 \oplus X_4$----------(3)

Where i is data word in x_i

$P_1 = X_1 \oplus X_5 \oplus X_9 \oplus X_{13}$----------(4)

$P_2 = X_2 \oplus X_6 \oplus X_{10} \oplus X_{14}$----------(5)

$P_3 = X_3 \oplus X_7 \oplus X_{11} \oplus X_{15}$----------(6)

$P_4 = X_4 \oplus X_8 \oplus X_{12} \oplus X_{16}$----------(7)

$Cx, y = Ax.By = \sum_{z-1}^{j} Ax, z\,.\,Bz, y$----------(8)

$\sum_{x-1}^{i}\sum_{y-1}^{k} Cx, y = \sum_{x-1}^{i}\sum_{y-1}^{k}\sum_{z-1}^{j} Ax, z\,.\,Bz, y$----------(9)

Using the above equation light ABFT cost analysis are calculations for algorithmic and architectural fault costs.

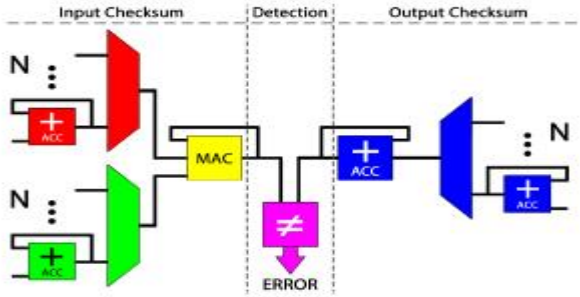


Fig 3.5 Error indication of MxM in Light ABFT systolic array architecture

Fig 3.5 shows the error indication of MxM in light ABFT systolic arrays. It indicates if the error is present in the MxM are not. It contains the input checksum, detection and the output checksum.

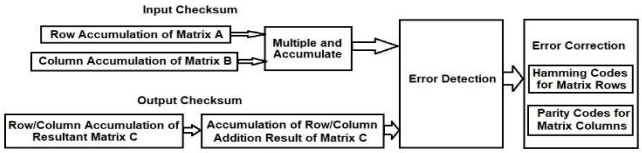


Fig 3.6 Proposed Error correction technique

Fig 3.6 shows the proposed Error correction technique. Using matrix multiplication, the systolic arrays are implemented on the Field-Programmable Gate Arrays using the lightweight ABFT. In the conventional ABFT requires summation vectors to input matrices i.e. a fault tolerance algorithm introduced by unnecessary computations (i+k+1) due to the expanded output matrix size.

To overcome this lightweight ABFT is introduced the computational overheads are reduced due to this. Instead of calculating i+k+1 light ABFT calculates just one element that intern reduces the redundant calculations. It compares the ik output elements from the array and compare it with pre-calculated checksum i.e. L. This reduces the hardware by -k+1 adders and a single comparator.

The efficiency of Light ABFT is high i.e. the light ABFT efficiently detects the errors. The system enters the safe state till it takes the corrective measures. When the larger multiplication exceeds the systolic array fixed size the computations are broken into smaller parts. To overcome this the blocking is done i.e. the operation is divided into square blocks of size B.

The alternative error correction technique like matrix codes that combines hamming codes and parity codes. To identify the multiple bit errors and high defect rates this offers the potential benefits.

The light ABFT is designed for error detection in matrix multiplication of FPGAs in systolic array.

X_1	X_2	X_3	X_4	C_1	C_2	C_3
X_5	X_6	X_7	X_8	C_4	C_5	C_6
X_9	X_{10}	X_{11}	X_{12}	C_7	C_8	C_9
X_{13}	X_{14}	X_{15}	X_{16}	C_{10}	C_{11}	C_{12}
P_1	P_2	P_3	P_4			

Figure 1: Check bits and parity bits of 16-Bit word

By combining hamming codes and vertical parity bits by enhancing error detection in a 16-bit word. The 4*4 matrix is restructured format of 16-bit word (k1=k2=k3). To each row of matrix hamming code is applied and C1,C2,C3 are the three check bits are added based on the hamming code implementation. By XOR operation these check bits are calculated with specific data, this is the horizontal detection of error using hamming code using equation 1. The vertical detection of error in parity bits for each column of matrix four vertical parity bit is added P1-P4. P is calculated using equation 2. The total number of 1s in the column is even or odd is

signified by the parity bits. These increases the error detection capabilities. It detects single and multi-bit errors using hamming code and single bit error are detected by parity bits.

$$C_1 = X_2 \oplus X_3 \oplus X_4$$
$$C_2 = X_1 \oplus X_3 \oplus X_4$$
$$C_3 = X_1 \oplus X_2 \oplus X_4$$
----------(1)

Where i is the data word in Xi.

$$P_1 = X_1 \oplus X_5 \oplus X_9 \oplus X_{13}$$
$$P_2 = X_2 \oplus X_6 \oplus X_{10} \oplus X_{14}$$
$$P_3 = X_3 \oplus X_7 \oplus X_{11} \oplus X_{15}$$
$$P_4 = X_4 \oplus X_8 \oplus X_{12} \oplus X_{16}$$
----------(2)

TABLE II COMPARISON OF ALGORITHMIC AND ARCHITECTURAL COST

Steps	Cost	Type	Existing method	Light ABFT
1	Algorithmic	ADD	j(i+k)	j(i+k)
	Architectural	ADD	2j	2j
3	Algorithmic	MUL ADD	j(i+k+1) j(i+k+1)	j j
	Architectural	MUL ADD	i+k+1 i+k+1	1 1
4	Algorithmic	ADD	2ik+i+k	ik+1
	Architectural	ADD	i+k+1	k+2
Total	Algorithmic	ADD MUL ADD	2(ik+ji+jk)+i+j+k ji+jk+j 2(i+j+k)+3	ik+ji+jk+j+1 j 2j+k+3
	Architectural	MUL	i+k+1	1
i=j=k	Algorithmic	ADD MUL ADD	$6M^2+3M$ $2M^2+M$ 6N+3	$3M^2+M+1$ M 3N+3
	Architectural	MUL	2N+1	1

Table II shows the comparison of the algorithmic and architectural cost of existing and the proposed system with adder and multiplier types.

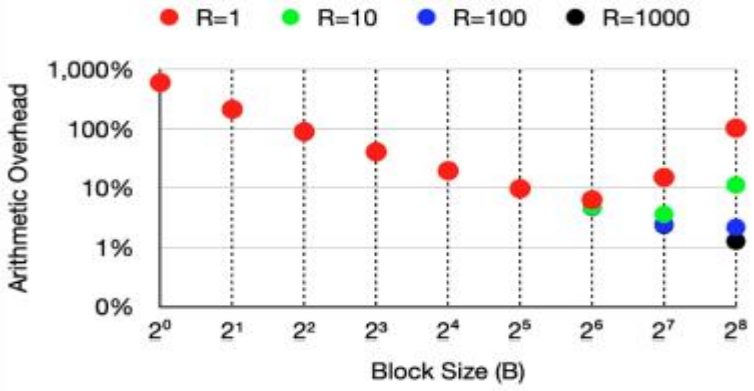


Fig 3.7 Arithmetic overhead and block size B for the matrix M=256

Fig 3.7 shows the arithmetic overhead and block size B for the matrix M=256.

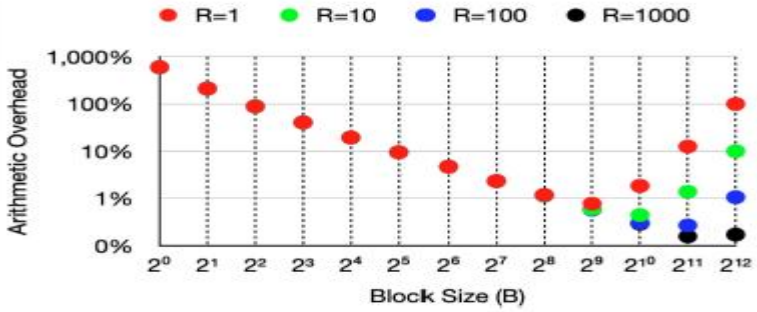


Fig 3.8 Arithmetic overhead and block size B for the matrix M=4096

Fig 3.8 shows the arithmetic overhead and block size B for the matrix M=4096.

TABLE III HAMMING CODE (7,4) ENCODING FORMULA

b_0	b_1	b_2	b_3	b_4	b_5	b_6	**Encoding formulas**
		u_0		u_1	u_2	u_3	
1	0	1	0	1	0	1	$b_0 = u_0 \oplus u_1 \oplus u_3$
0	1	1	0	0	1	1	$b_1 = u_0 \oplus u_2 \oplus u_3$
0	0	0	1	1	1	1	$b_3 = u_1 \oplus u_2 \oplus u_3$

Table III shows the hamming code (7,4) encoding formula with output of b_0, b_1, b_3 with XOR operation performed with u_0, u_1, u_2, u_3.

TABLE IV HAMMING CODE (7,4) SYNDROME BITS

r_0	r_1	r_2	r_3	r_4	r_5	r_6	**Syndrome bits**
		u_0		u_1	u_2	u_3	
1	0	1	0	1	0	1	$S_0 = r_0 \oplus r_2 \oplus r_4 \oplus r_6$
0	1	1	0	0	1	1	$S_1 = r_1 \oplus r_2 \oplus r_5 \oplus r_6$
0	0	0	1	1	1	1	$S_1 = r_1 \oplus r_2 \oplus r_5 \oplus r_6$

Table IV shows the hamming code (7,4) encoding formula with output of S_0, S_1 with XOR operation performed with r_0, r_1, r_2, r_3, r_4, r_5, r_6.

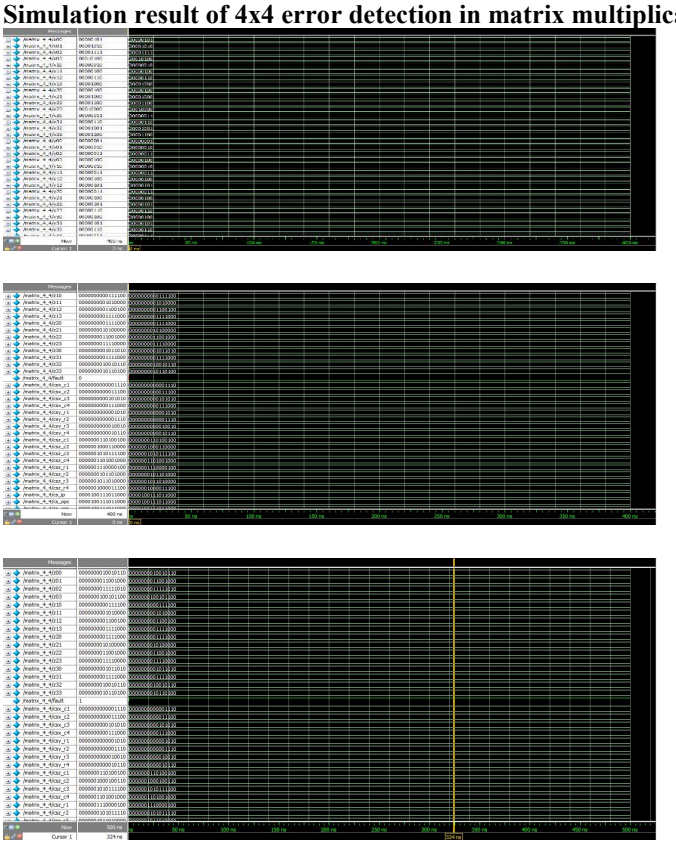
TABLE V ERROR DETECTION AND CORRECTION WITH EXTENDED HAMMING CODES

Syndrome bits	Parity whole word	Event	Action
Zero	Even	No Error	-
Zero	Odd	Single Error (in the parity bit)	-
Nonzero	Even	Two-Bit-Error (non-recoverable error	Disable correction logic signal the error
Nonzero	Odd	Single Error (in bits 0 to 6)	Enable correction logic

Table V shows the error detection and correction with extended hamming codes. Which contains zero and non-zero bits of syndrome bits. Parity whole word of even and odd. The event of no error and single error in the parity bit and two-bit error of non-recoverable error and single error in bits 0 to 6 with action of disable correction logic signal the error and enable correction logic.

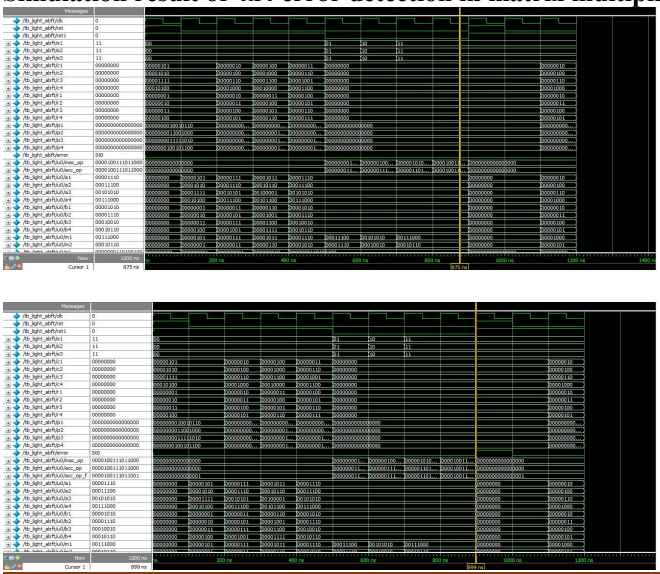
IV. EVALUATION RESULTS

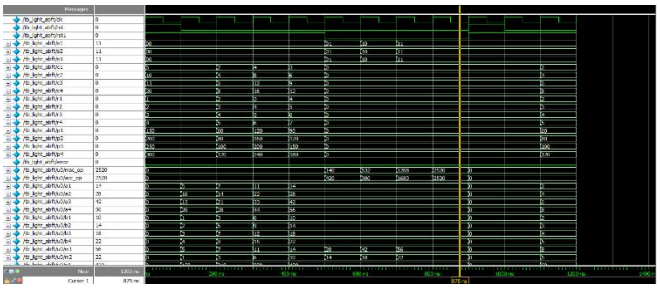
Simulation result of 4x4 error detection in matrix multiplication using traditional ABFT

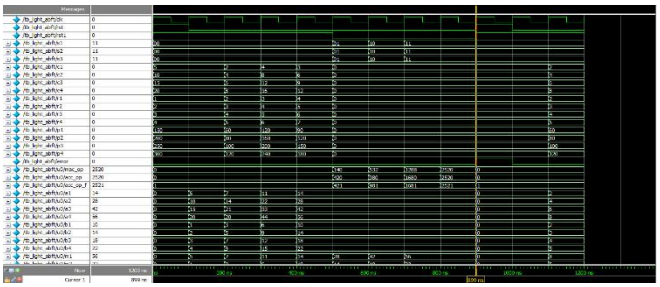


X00 to X33 and Y00 to Y33 is the input 4x4 matrices used for matrix multiplication. Z00 to Z33 is the output of the 44-matrix multiplication. FAULT signal represents the fault in matrix multiplication by comparing the checksum. Fault value is '0' if there is no fault else fault signal becomes'1'. ABFT algorithm lies in the data redundancy before matrix multiplication. the summation vectors are introduced to input matrices which enables the error detection and correction.

Simulation result of 4x4 error detection in matrix multiplication using LIGHT ABFT

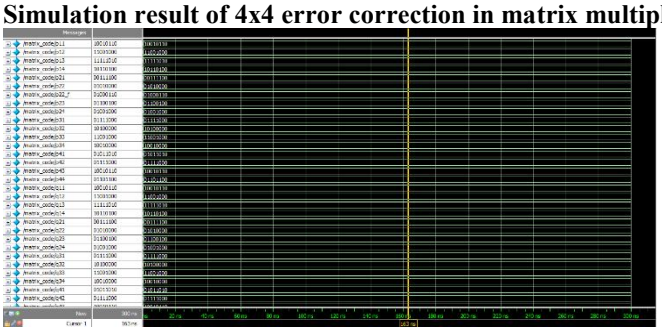


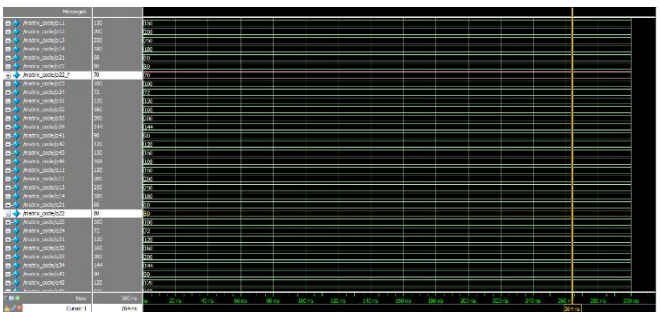




C1-C4 is the column input of the input matrix A and R1-R4 is the row input of the input matrix B with P1-P4 is the input of the output matrix. Instead of performing multiple accumulation and comparisons it accumulates all output elements ik from the array are compared with the sum to a single checksum (L) reduces -k+1 adders and single comparator.

Simulation result of 4x4 error correction in matrix multiplication using LIGHT ABFT





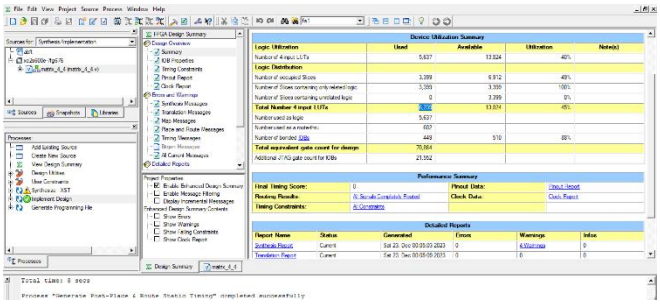
P11-P44 is the resultant output matrix. If the error is present in MxM computation the unit outputs the results if the unit enters the input from left and output from right. Vertical parity bits P1-P4 are calculated using vertical XOR operation. The simulation shows the binary and decimal format results. P22_F indicates the faulty element and it is corrected and obtained in Q22.

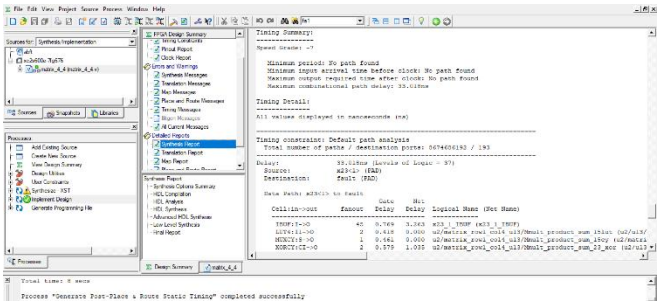
TABLE VI COMPARISON TABLE

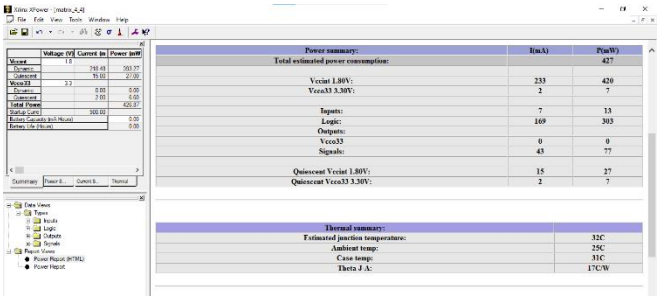
Algorithm	Gate count	LUTs	Slices	Delay(ns)	Power (mW)
ABFT	70864	6239	3399	33.018	426.87
Light ABFT	4452	303	157	14.730	247.20

Table VI shows the comparison table of ABFT and light ABFT. The gate count of 4452 which is less than ABFT. The LUTs is of 303 and delay of 14.730ns and total consumed power of 247.20 milliwatt.

ABFT



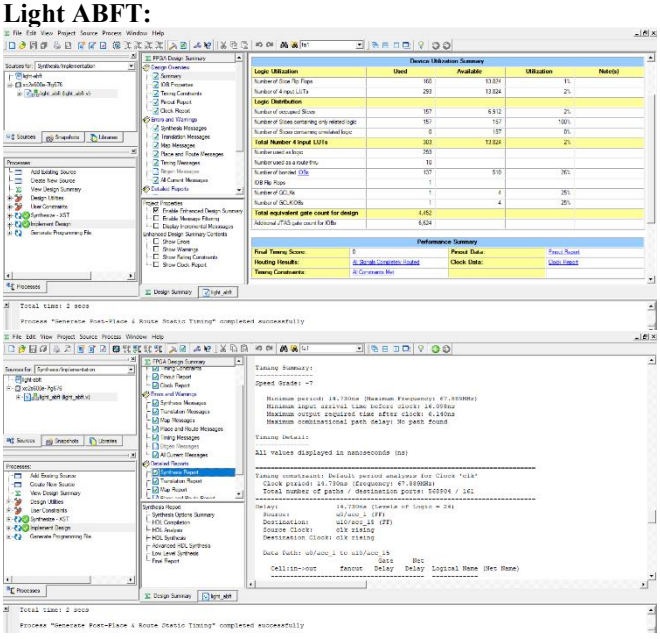


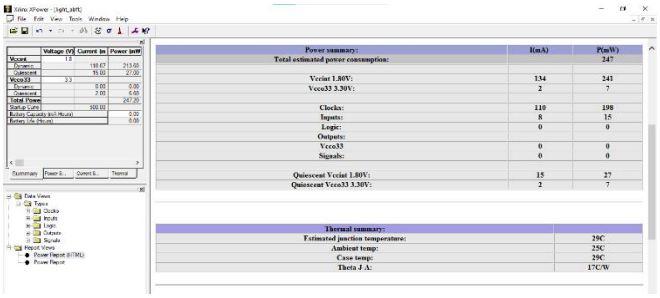


Power summary:	I(mA)	P(mW)
Total estimated power consumption:		427
Vccint 1.80V:	233	420
Vcco33 3.30V:	2	7
Inputs:	7	13
Logic:	169	303
Outputs:		
Vcco33	0	0
Signals:	43	77
Quiescent Vccint 1.80V:	15	27
Quiescent Vcco33 3.30V:	2	7

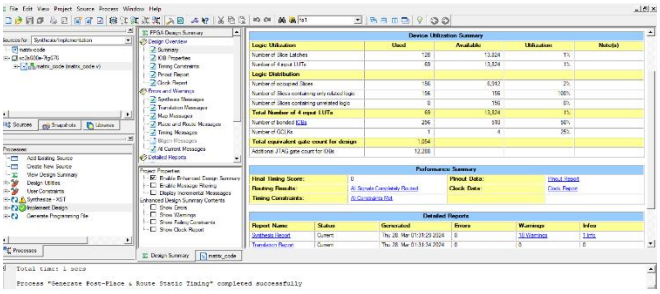
Thermal summary:	
Estimated junction temperature:	32C
Ambient temp:	25C
Case temp:	31C
Theta J-A:	17C/W

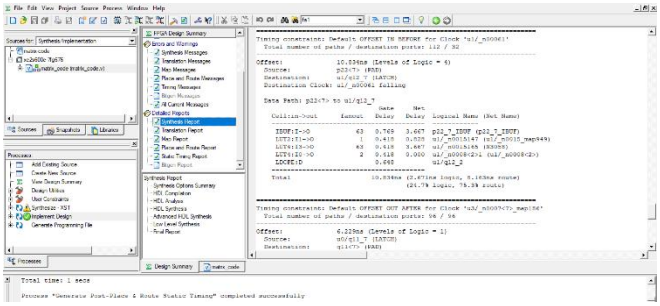
Light ABFT:

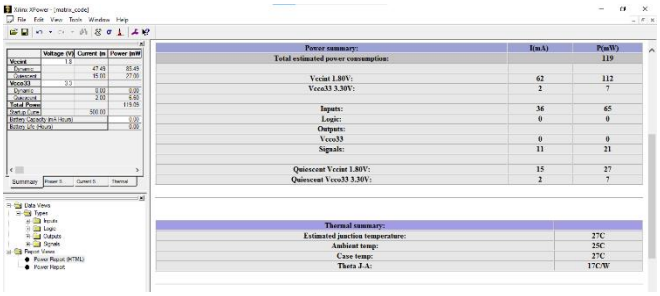




Method	Gate count	LUTs	Slices	Delay(ns)	Power (mW)
Matrix code	1054	69	156	10.834	119.09







V. CONCLUSION

The high performance and the reliable matrix multiplication is achieved through light ABFT by the implementation of cost-effective error detection technique at various levels by selecting parallel processing architecture. The systolic array implementation of FPGAs is done with the light ABFT which increases the solution for error detection . Comparing to conventional methods the proposed matrix code offers an enhanced efficiency and increased performance. The speed and reliability of matrix multiplication is enhanced using combined strategy.

REFERENCES

[1] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in Proc. IEEE 36th VLSI Test Symp., 2018, pp. 1–6.

[2] A. Ruospo, A. Bosio, A. Ianne, and E. Sanchez, "Evaluating convolutional neural networks reliability depending on their data representation," in Proc. 23rd Euromicro Conf. Digit. Syst. Des., 2020, pp. 672–679.

[3] Xilinx, "UltraScale architecture DSP slice user guide. xilinx," 2019. [Online]. Available: https://www.xilinx.com/support/documentation/user_ guides/ug579-ultrascale-dsp.pdf
[4] E. Keren, S. Greenberg, N. M. Yitzhak, D. David, N. Refaeli, and A. Haran, "Characterization and mitigation of single-event transients in xilinx 45-nm SRAM-Based FPGA," IEEE Trans. Nucl. Sci., vol. 66, no. 6, pp. 946–954, Jun. 2019.
[5] Xilinx. DPU. Zynq Xilinx. 2019. [Online]. Available: https://www. xilinx.com/support/documentation/ip_documentation/dpu/v3_1/pg338- dpu.pdf
[6] S. K. S. Hari, M. Sullivan, T. Tsai, and S. W. Keckler, "Making convolutions resilient via algorithm-based error detection techniques," IEEE Trans. Dependable Secure Comput., vol. 19, no. 4, pp. 2546–2558, Jul./Aug. 2022.
[7] T. Marty, T. Yuki, and S. Derrien, "Enabling overclocking through algorithm-level error detection," in Proc. Int. Conf. Field-Programmable Technol., 2018, pp. 174–181.
[8] F. F. d. Santos et al., "Analyzing and increasing the reliability of convolutional neural networks on GPUs," IEEE Trans. Rel., vol. 68, no. 2, pp. 663–677, Jun. 2019.
[9] L. K. Draghetti, F. F. d. Santos, L. Carro, and P. Rech, "Detecting errors in convolutional neural networks using inter frame spatio-temporal correlation," in Proc. IEEE 25th Int. Symp. On-Line Testing Robust Syst. Des., 2019, pp. 310–315.
[10] L. A. Aranda, P. Reviriego, and J. A. Maestro, "A comparison of dual modular redundancy and concurrent error detection in finite impulse response filters implemented in SRAM-Based FPGAs through fault injection," IEEE Trans. Circuits Syst. II: Exp. Briefs, vol. 65, no. 3, pp. 376–380, Mar. 2018.
[11] R. González-Toral, P. Reviriego, J. A. Maestro, and Z. Gao, "A scheme to design concurrent error detection techniques for the fast fourier transform implemented in SRAM-Based FPGAs," IEEE Trans. Comput., vol. 67, no. 7, pp. 1039–1045, Jul. 2018.
[12] M. Qasaimeh, J. Zambreno, P. H. Jones, K. Denolf, J. Lo, and K. Vissers, "Analyzing the energy-efficiency of vision kernels on embedded CPU, GPU and FPGA platforms," in Proc. IEEE 27th Annu. Int. Symp. Field-Programmable Custom Comput. Mach., 2019, pp. 336–336.
[13] Top500.org, "The linpack benchmark," 2020. [Online]. Available: https: //www.top500.org/project/linpack/
[14] Tesla. Autopilot. Accessed: Feb. 25, 2023. [Online]. Available: https:// www.tesla.com/autopilot
[15] H. R. Kerner, K. L. Wagstaff, B. D. Bue, P. C. Gray, J. F. Bell, and H. Ben Amor, "Toward generalized change detection on planetary surfaces with convolutional autoencoders and transfer learning," IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 12, no. 10, pp. 3900–3918, Oct. 2019.