International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

An Efficient Design of Fault-Tolerance for Matrix–Vector Multiplications

P. Benesh Selva Nesan[1], S.Soban[2]

[1]*Assistant Professor of ECE, ROHINI COLLEGE OF ENGINEERING & TECHNOLOGY*
[2]*Assistant Professor of ECE, ROHINI COLLEGE OF ENGINEERING & TECHNOLOGY*

Abstract — *Equal lattice handling is a run of the mill activity in numerous frameworks, and specifically grid vector augmentation (MVM) is one of the most widely recognized tasks in the advanced computerized signal preparing and advanced correspondence frameworks. This paper proposes a fault tolerant plan for number equal MVMs. The plan consolidates thoughts from mistake rectification codes with oneself checking capacity of MVM. Field-programmable entryway exhibit assessment shows that the proposed plan can fundamentally decrease the overheads contrasted with the security of each MVM all alone. In this way, the proposed strategy can be utilized to lessen the expense of giving adaptation to internal failure in down to earth usage.*

Key Words: Fault tolerance, matrix–vector multiplications (MVMs), parallel processing, soft errors.

1. INTRODUCTION

Matrix Vector Multiplication (MVM) is an average calculation in space change, projection, or highlight extraction [1], [2]. With expanding interest for elite or throughput, equal registering is getting progressively significant in superior figuring frameworks, distributed computing frameworks, and correspondence frameworks [3], [4]. For instance, equal lattice handling with a typical information vector (equal MVMs) is performed for precoding in multiuser multi-in multi-out (MIMO) (particularly enormous scope MIMO) [5]–[7] and elite low thickness equality check decoders [8], [9]. Equal handling may run on countless preparing units, e.g., GPUs or disseminated frameworks, and investigations show that delicate mistakes can influence the yields [10], [11].Adaptation to non-critical failure for the network handling has generally been examined [11]–[17]. References [11]–[14] center around calculation based faulttolerance (ABFT) methods for framework lattice augmentation. In this ABFT structure, the two info grids are changed to a "line checksum" framework and a "section checksum" network, separately, and single blunders in the subsequent "full checksum" lattice can be distinguished and rectified dependent on its line checksum and segment checksum [12]. Reference [15] gives an overall model to such procedures. Since the line checksum doesn't exist for an info vector, such ABFT plan must be utilized for mistake location in MVM. In the ABFT schemes proposed in [16], checksum technique is used to locate the affected result element or blocks of result elements in MVM, and partial recomputation is used for error correction. Such schemes are suitable for the cases in which the delay introduced by recomputation is acceptable. But this is not always the case, especially in high-throughput communication receivers where the inputs arrive in real time from A/D converters. A general method to protect linear dynamic systems was proposed. Part of that system model can be extracted as an MVM. A method to protect parallel digital filters was proposed. The scheme is based on considering each filter as a bit in an error correction code (ECC) so that parity check filters are added and used to detect and correct errors.

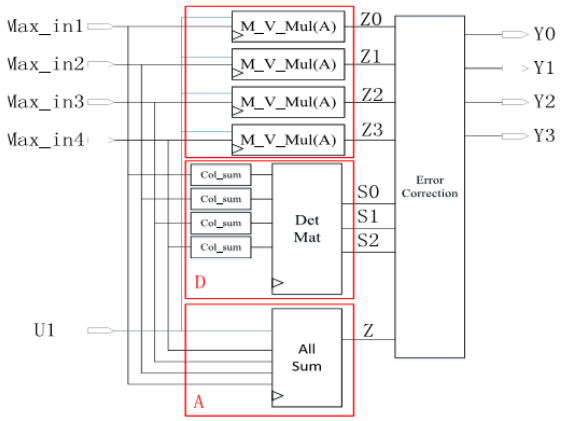

Fig-1 STRUCTURE OF SCHEME 2 PARALLEL MVM

2. METHODOLOGIES

In a recent technology of Digital Signal processing application method, will increasing demand in high performance computing system, cloud computing and communication systems, in this method a multiplication is highest priority in all arithmetic process, due to this multiplication, a vector and matrix vector multiplication is one of a typical computation domain regarding

transformation, projection and feature extraction, and its perform pre-coding method in multiuser multi-in multi-out(MIMO) applications. In this method of Matrix Vector Multiplication, a fault tolerance error will occur on the time of transmission and reception, respectively. Here, a proposed work will have reduce this error and overcome a method of single fault error correction to multiple fault error correction with considered BCH Code, it have capable to detect and correct a Multiple Fault correction , and it will use matrices to extend idea from Hamming and Reed-Solomon Codes, with detect many error in MVM. In this method to implement in VHDL and Synthesized in Xilinx FPGA S6LX9, finally compared with Hamming method of Single fault error correction and shown in terms of area, delay and power.

2.1 Implementation of the Matrix–Vector Multiplication

The structure of the MVM is shown in Fig. 2. A sequential procedure is applied to resource efficiency. For each cycle, a column of matrix Ap is read into the data registers (a0 to a19 in Fig. 4), and the product with the corresponding item in u^-(ui in Fig. 2) is calculated. A cycle counter is used to control a group of selectors. Before the counter reaches M, port 1 of each selector is selected so that the result is accumulated. When the counter reaches M, port 0 of each selector is selected to output the result vector. For MVM between an N × M matrix and an M ×1 vector, N multipliers and N adders are needed in the structure, and an N × 1 vector is generated every M cycles. The results of the MVMs under protection are 21-bit integers.

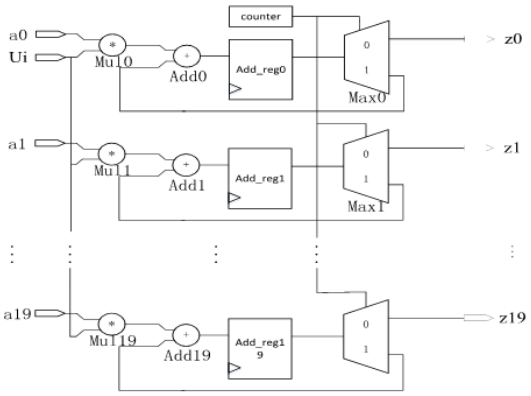

Fig-2 Implementation structure of MVM (N=20)

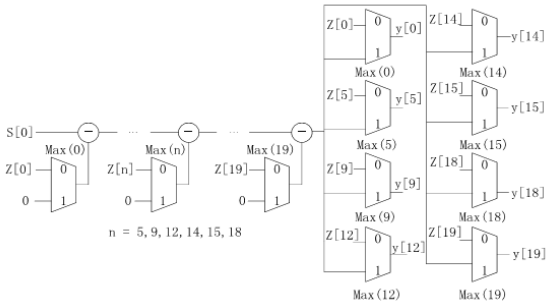

Fig-3 Structure of fault recovery for separate protection

2.2 Implementation of the Proposed Scheme

The implementation structure of the proposed scheme for P parallel MVMs is shown in Fig. 4 (P=4). The calculation of D^- u (D is a 3 ×30 matrix) and A^- u (A is a 20 ×30 matrix) also apply the MVM module in Fig. 4. For each cycle, one column for each of the four matrices (Max_in1 to Max_in4) and the corresponding item (ui) from vector $^-$ u is read into the module.

For the detection branch, each of the four input columns are accumulated by the Col_Sum module, getting c^1_i, c^2_i, c^3_i and c^4_i (13-bit integers). Then the i_{th} column (3 ×1) of the detection matrix D is generated according to a Hamming code. The items in D are 15-bit integers, and the items in the result vector of D^- u are 28-bit integers.

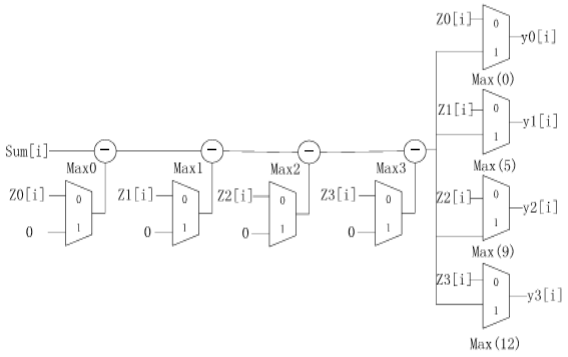

Fig-4 Implementation Structure for fault recovery in the proposed system

2.3 BCH codes

BCH codes use variant check matrices to extend ideas from Hamming and Reed-Solomon codes. BCH codes can keep a fixed alphabet, and correct many errors. Unfortunately, BCH codes suffer from the same shortcoming as Hamming and RS codes: as block size increases, they become worse and worse in the sense that the relative error correction rate goes to 0 (and/or information rate goes to 0). Start with a finite field Fq with q elements, often q = 2. Choose block size n, for simplicity relatively prime to q. For q = 2, look at odd block sizes.

RESULT AND DISCUSSION

The separate protection (or scheme 1) and the proposed scheme (scheme 2) are implemented for 4 and 8 parallel MVMs, respectively, using Verilog in Vivado2014.4 for Xilinx xc7vx485tffg1157-1 with option of area optimization. In particular, each matrix has 20 rows (N = 20) and 30 columns (M = 30), and both the matrix and vector items are 8-bit integers. Both the matrix items and the vector items are uniformly distributed between −128 and 127. All the intermediate and final results are assigned enough bits so that no rounding is applied during the operations.

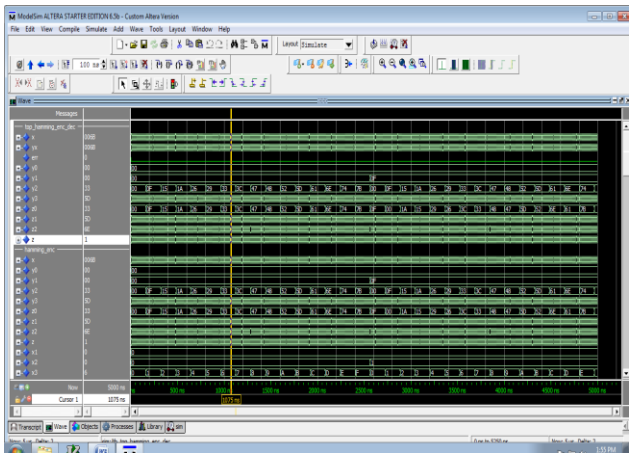

Fig-5 Hamming output

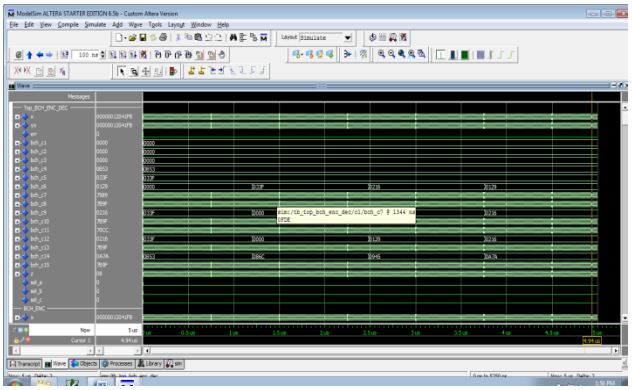

Fig-6 BCH output

	LUT	Occupied Slice	IOB	Fanout	Power(mw)	Delay(ns)
Hamming	242	130	33	3.89	25	25.993
BCH	1403	732	99	3.75	25	63.479

Table-1 Comparison between existing and proposed system

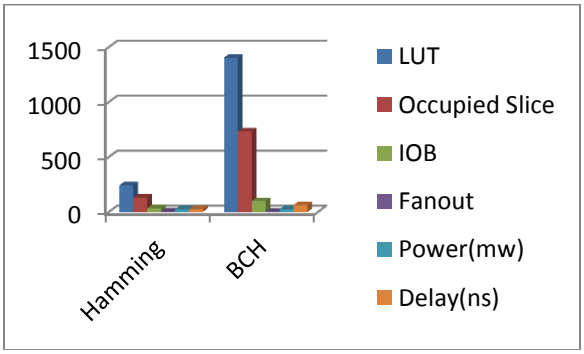

Fig-7 Performance graph

CONCLUSION

In this paper, a fault tolerant plan for equal MVMs was proposed. The plan joins the lattice self-checking (checksum) and the utilization of the mistake remedy coding. FPGA-based assessment shows that the proposed plan can lessen the overhead required by up to 20% and 40% contrasted with discrete assurance, for 4 and 8 equal MVMs, separately. This preferred position would be bigger for progressively equal branches. In spite of the fact that this paper just exhibits the plan for the case that only one MVM comes up short, it very well may be stretched out to

the case that different MVMs bomb by choosing the codes with bigger separation.

REFERENCES

[1] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[2] J. S. Lim, Two-Dimensional Digital Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
[3] A. Grama, A. Gupta, G. Karypis, and V. Kumar, Introduction to Parallel Computing, 2nd ed. London, U.K.: Pearson, Jan. 2003.
[4] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Hoboken, NJ, USA: Wiley, 1999.
[5] S. Roger, C. Ramiro, A. Gonzalez, V. Almenar, and A. M. Vidal, "Fully parallel GPU implementation of a fixed-complexity soft-output MIMO detector," IEEE Trans. Veh. Technol., vol. 61, no. 8, pp. 3796–3800, Oct. 2012.
[6] Q. Qi and C. Chakrabarti, "Parallel high throughput soft-output sphere decoder," in Proc. IEEE Workshop Signal Process. Syst., Oct. 2010, pp. 174–179.
[7] M. Wu, C. Dick, and J. R. Cavallaro, "Improving MIMO sphere detection through antenna detection order scheduling," in Proc. SDR Forum, 2011, pp. 280–284.
[8] G. Wang, M. Wu, Y. Sun, and J. R. Cavallaro, "A massively parallel implementation of QC-LDPC decoder on GPU," in Proc. IEEE 9th Symp. Appl. Specific Process. (SASP), Jun. 2011, pp. 82–85.
[9] G. Wang, H. Shen, B. Yin, M. Wu, Y. Sun, and J. R. Cavallaro, "Parallel nonbinary LDPC decoding on GPU," in Proc. ASILOMAR Conf., Nov. 2012, pp. 1277–1281.
[10] I. S. Haque and V. S. Pande, "Hard data on soft errors: A largescale assessment of real-world error rates in GPGPU," in Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput. (CCGrid), May 2010, pp. 691–696.
[11] P. Rech, C. Aguiar, C. Frost, and L. Carro, "An efficient and experimentally tuned software-based hardening strategy for matrix multiplication on GPUs," IEEE Trans. Nucl. Sci., vol. 60, no. 4, pp. 2797–2804, Aug. 2013.
[12] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," IEEE Trans. Comput., vol. C-33, no. 6, pp. 518–528, Jun. 1984.
[13] C. Ding, C. Karlsson, H. Liu, T. Davies, and Z. Chen, "Matrix multiplication on GPUs with on-line fault tolerance," in Proc. IEEE 9th Int. Symp. Parallel Distrib. Process. Appl., May 2011, pp. 311–317.
[14] P. Wu, C. Ding, L. Chen, T. Davies, C. Karlsson, and Z. Chen, "On-line soft error correction in matrix–matrix multiplication," J. Comput. Sci., vol. 4, no. 6, pp. 465–472, Nov. 2013.
[15] C. J. Anfinson and F. T. Luk, "A linear algebraic model of algorithmbased fault tolerance," IEEE Trans. Comput., vol. C-37, no. 12, pp. 1599–1604, Dec. 1988for liver segmentation and lesions detection,'' in Deep Learning and Data Labeling for Medical Applications (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2016, pp. 77–85.
[16] J. Sloan, R. Kumar, and G. Bronevetsky, "An algorithmic approach to error localization and partial recomputation for low-overhead fault tolerance," in Proc. 43rd IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), Jun. 2013, pp. 1–12.