International Journal For Applied Research in Science and Engineering
Volume 4, Issue 5, December-2021 *ISSN (Online): 2456-124X*

AN EXTENSIVE SURVEY OF LITERATURE ON EFFICIENT FAULT-TOLERANCE DESIGN FOR INTEGER PARALLEL MATRIX–VECTOR MULTIPLICATIONS

Kailash Chakradhari[1], Prof. Balram Gupta[2]
[1]Mtech Scholar, [2]Guide
SCOPE college of engineering Bhopal

Abstract- ***Matrix multiplication is widely used as core operation in various signals processing application like software defined radio. The FFT processor is widely used in DSP and communication applications. Recently, both high data processing and high power efficiency consumes more power. Due to the nature of non-stop processing at the sample rate, the pipelined FFT appears to be the leading architecture for high performance applications. Since these two functions are widely used in various mobile devices they required to have features like low power, lesser area without increase of latency. With ever growing reliance on computers and proliferation of computer services, the need for dependable computing systems has become very crucial. Fault tolerance is a design attribute of a system that enables fault tolerant and dependable functioning of a system in spite of failures in the system. This examination reported an extensive review on Fault-Tolerance Design for Integer Parallel Matrix–Vector Multiplications.***

Keywords- ***Matrix–Vector Multiplications, FPGA, Fault-Tolerance Circuit, VLSI, Parallel Design.***

1. INTRODUCTION

Matrix multiplication and Fast Fourier Transform are important tools used in the Digital Signal Processing applications. Each of them is compute-intensive portion of broadband beam forming applications such as those generally used in software defined radio and sensor networks. These are frequently used kernel operations in signal and image processing systems including mobile systems.

Recently, in signal processing there has been a lot of development to increase its performance both at the algorithmic level and the hardware implementation level. Researchers have been developing efficient algorithms to increase the speed and to keep the memory size low. On the other hand, developers of the VLSI systems are including features in design that improves the system performance for applications requiring matrix multiplication and Fast Fourier Transform. Research in this field is not only because of the popularity, but also because of the reason that, for decades the chip size has decreased drastically. This has allowed portable systems to integrate more functions and become more powerful. These advances have also, unfortunately, led to increase in power consumption. This has resulted in a situation, where numbers of potential applications are limited by the power - not the performance. Therefore, power consumption has resulted to be the most significant design requirement in portable systems and this has lead to many low power design techniques and algorithms.

Matrix-Vector Multiplication is directly related to a wide variety of computational disciplines, including circuit and economic modeling, industrial engineering, image reconstruction, algorithms for least squares and eigenvalue problem. Floating-point Matrix-Vector Multiplication, generally denoted as y = Ax, is the key computational kernel that dominates the performance of many scientific and engineering applications. However, the performance of Matrix-Vector Multiplication is highly limited by the irregularity of memory accesses as well as high ratio of memory I/O operations, and is usually much lower than that of dense matrix multiplication.

Many real life numerical problems in applications such as engineering simulations, scientific computing, information retrieval, and economics use matrices where there are few interactions between elements and hence most of the matrix entries are zero. For these common problems, dense matrix representations and algorithms are inefficient. There is no reason to store the zero entries in memory or to perform computations on them. Consequently, it is important to use sparse matrix representations for these applications. The sparse matrix representations only explicitly represent non-zero matrix entries and only perform operations on the non-zero matrix elements. Further, sparse parallel algorithms often take advantage of matrix locality to perform much less communication on parallel machines than their dense counter parts.

The advantage of FPGAs is that many floating point operations can be performed in parallel. The parallelization strategy chooses must allow use of many Processing Elements (PE). Parallelize over the set of dot products in the matrix multiply, assigning a minimum of one dot product to each PE. So the maximum usable number of PEs is the dimension of the matrix. Further scaling would require breaking dot products between processing elements. Parallel scaling can also be limited by the large amount of communication work required when there are many PEs.

For many approaches, large communication work between processing elements is the main scaling limiter. Perform offline data placement to minimize communication by exploiting locality in the sparse matrix. The interconnect for design is a bidirectional ring which allows inexpensive local communication between PEs.

2. MATRIX VECTOR MULTIPLICATION AND FAULT TOLERANCE

A. Sparse Matrix Sparse Vector Multiplication

Sparse Matrix Sparse Vector Multiplication is simply the

multiplication of a sparse matrix by a sparse vector. The general format follows typical matrix vector multiplication except that it would be a waste of time to multiply zeros by any number. Figure illustrates this dilemma.

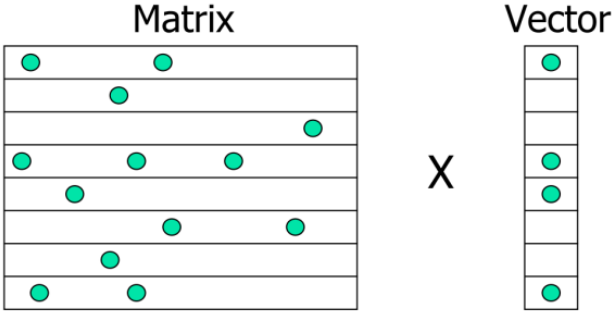

Fig.2.1 Sparse Matrix Sparse Vector Multiplication

To handle this implementation, a storage scheme is used to hold the data for the sparse structures. Due to the storage scheming, matrix vector multiplication is no longer a straightforward operation. The column address of the current row of a matrix being multiplied must correspond with an existing row address of the vector. If there is a match, then the two corresponding values can be multiplied together.

The matrix value's column address must be compared to the vector's row address and cannot be directly mapped as above. If the matrix address is less than the vector address, then the next matrix value's address needs to be compared. If the matrix value's address is greater than the vector address', then the next vector value's address must be retrieved. If they both match, then they are obviously multiplied together address', then the next vector value's address must be retrieved. If they both match, then they are obviously multiplied together.

B. Fault Methodologies

There are various fault tolerant approaches for FPGAs that have been proposed throughout the years. Approaches reach out from basic incorporation of additional segments that are not entirely important to working, in the event of failure in different segments to completely online versatile executions.

A fault can be defined as a defect or unsatisfactory condition of a system, which may lead to failure, or illogical function of the system. Fault tolerance is the methodology of the hardware system that in case of occurrence of any logical error or hardware faults, the correct logical working of the system can still be achieved by having an alternative backup procedure to immediately take up its place to maintain the precise, error-free functionality of the system. One of the distinct advantages of using reconfigurable devices, such as FPGAs, is that the current functionality of the device can be altered or changed sometime in the future. This supplementary benefit allows the designers to completely reprogram the FPGA with a new logic.

The basic principle of design for fault tolerance is to provide the system with some extra resources / redundancy (hardware, software, run time, additional information etc.) beyond what is required to accomplish the given tasks and use these extra resources to mask , overcome, the effect of a malfunction autonomously , without human intervention. It is a fault management technique which negates the effect of failures.

Besides ultra high reliability, the need for fault tolerance is driven by other factors such as ultra high availability reduced life cycle costs and long life and unattended operations. Fault tolerant computers are also called as : resilient computers, 'computers that can't fail ' , 'nonstop systems' , ' no down time computers', 'zero defect systems' and crash resistant computers.

Design of fault tolerance is a broad discipline, the diverse areas of which range from study of failure mechanisms in integrated circuits to the design of robust software. It requires a good understanding of a large and complex set of interrelated subjects.

3. LITERATURE REVIEW

SR. NO.	TITLE	AUTHOR	YEAR	APPROACH
1	An Efficient Fault-Tolerance Design for Integer Parallel Matrix–Vector Multiplications	Z. Gao, Q. Jing, Y. Li, P. Reviriego and J. A. Maestro	2018	A fault-tolerant design for integer parallel MVMs has reported using FPGA
2	FPGA implementation of matrix-vector multiplication using Xilinx System Generator	I. Sayahi, M. Machhout and R. Tourki	2018	Reported a new FPGA design and implementation for matrix vector multiplication
3	A bandwidth in-sensitive low stall sparse matrix vector multiplication architecture on reconfigurable FPGA platform	S. M. Ali, W. Shaojun, M. Ning and P. Yu	2017	FPGA optimized single bit stream column major sparse compression format that ensures decreased memory bandwidth and storage requiremen
4	Efficient fault tolerant parallel matrix-vector multiplications,	Z. Gao, P. Reviriego and J. A. Maestro,	2016	A fault tolerant design for parallel matrix-vector multiplications has reported
5	Efficient Algorithm-Based Fault Tolerance for Sparse Matrix Operations	A. Schöll, C. Braun, M. A. Kochte and H. Wunderli	2016	Reported a fault tolerance approach for sparse matrix operations that detects and implicitly locates errors in the results for efficient local correction.
6	Fast Sparse Matrix and Sparse Vector Multiplication Algorithm on the GPU	C. Yang, Y. Wang and J. D. Owens	2015	An efficient k-way merge lies at the heart of finding a fast parallel SpMSpV algorithm
7	Reducing Vector I/O for Faster GPU Sparse Matrix-Vector Multiplication	P. N. Q. Anh, R. Fan and Y. Wen	2015	Introduce two techniques to significantly decrease the I/O for vector accesses, by making novel use of the GPU's fast shared memory
8	No zero padded sparse matrix-vector multiplication on FPGAs	J. Huang, J. Ren, W. Yin and L. Wang	2014	Introduce a novel compressed element storage (CES) format, in which the additional data structures for indexing are abandoned

Z. Gao, Q. Jing, Y. Li, P. Reviriego and J. A. Maestro [1] Parallel matrix processing is a typical operation in many systems, and in particular matrix-vector multiplication (MVM) is one of the most common operations in the modern digital signal processing and digital communication systems. This examination work reported a fault-tolerant design for integer parallel MVMs. The scheme combines ideas from error correction codes with the self-checking capability of MVM. Field-programmable gate array evaluation shows that the proposed scheme can significantly reduce the overheads compared to the protection of each MVM on its own.

Therefore, the proposed technique can be used to reduce the cost of providing fault tolerance in practical implementations.

I. Sayahi, M. Machhout and R. Tourki [2] In diverse domains, the scientific applications requires severe computing algebra routines. The matrix multiplication presents an indispensable mathematical operation in many high performance fields. This work presents a new FPGA design and implementation for matrix vector multiplication. The design has been implemented with Xilinx System Generator. The results of FPGA implementation were compared with similar work on VIRTEX 4 platform. It demonstrates the efficiency of our work in term of resources utilization and speed up.

S. M. Ali, W. Shaojun, M. Ning and P. Yu [3] Sparse Matrix with dense Vector Multiplication (SpMxV) is a key computational kernel for most mathematical high performance computing problems. SpMxV implementation on FPGAs provides better performance but memory bandwidth limitations and pipeline stalling are still ongoing research concerns. This research proposes an FPGA optimized single bit stream column major sparse compression format that ensures decreased memory bandwidth and storage requirement. The architecture ensures low pipeline stalling by effectively performing 3 floating point operations per cycle per processing element. The proposed architecture is demonstrated with known benchmarks on ZYNQ-ZC702 FPGA. Result shows that on average our approach achieves compression ratios of 3.0 to 5.3 with respect to conventional CSR format. And peak compression ratio of 20.0 can be achieved for certain matrices used in our research. Low stall architecture proves 40 to 150% depreciation in stall cycles providing better overall efficiency.

Z. Gao, P. Reviriego and J. A. Maestro [4] Parallel matrix processing is a typical operation in many systems, and in particular matrix-vector multiplication is one of the most common operations in modern digital signal processing and digital communication systems. This work proposes a fault tolerant design for parallel matrix-vector multiplications. The scheme combines ideas from Error Correction Codes with the self-checking capability of matrix-vector multiplication.

A. Schöll, C. Braun, M. A. Kochte and H. Wunderlich, [5] this examination propose a fault tolerance approach for sparse matrix operations that detects and implicitly locates errors in the results for efficient local correction. This approach reduces the runtime overhead for fault tolerance and provides high error coverage. Existing algorithm-based fault tolerance approaches for sparse matrix operations detect and correct errors, but they often rely on expensive error localization steps. General checkpointing schemes can induce large recovery cost for high error rates. For sparse matrix-vector multiplications, experimental results show an average reduction in runtime overhead of 43.8%, while the error coverage is on average improved by 52.2% compared to related work. The practical applicability is demonstrated in a case study using the iterative Preconditioned Conjugate Gradient solver. When scaling the error rate by four orders of magnitude, the average runtime overhead increases only by 31.3% compared to low error rates.

C. Yang, Y. Wang and J. D. Owens, [6] this examination work implement a promising algorithm for sparse-matrix sparse-vector multiplication (SpMSpV) on the GPU. An efficient k-way merge lies at the heart of finding a fast parallel SpMSpV algorithm. This work examine the scalability of three approaches -- no sorting, merge sorting, and radix sorting -- in solving this problem. For breadth-first search (BFS), achieve a 1.26x speedup over state-of-the-art sparse-matrix dense-vector (SpMV) implementations. The algorithm seems generalize able for single-source shortest path (SSSP) and sparse-matrix sparse-matrix multiplication, and other core graph primitives such as maximal independent set and bipartite matching.

P. N. Q. Anh, R. Fan and Y. Wen, [7] Sparse matrix-vector multiplication (SpMV) is an important kernel used in solving many scientific and engineering problems. The massive parallelism of graphics processing units (GPUs) makes them well suited for SpMV computations. However, fully utilizing the power of GPUs is challenging because SpMV makes a large number of scattered memory accesses which saturate the GPU's memory bandwidth. Most previous works sought to address the bandwidth limitation by using efficient storage formats for the matrix. However, show that for most matrices, a majority of the bandwidth is consumed by accesses to the vector. In this examination, introduce two techniques to significantly decrease the I/O for vector accesses, by making novel use of the GPU's fast shared memory. A key advantage of our vector optimizations is that they are complementary to existing matrix I/O optimizations, so that it is possible to use both techniques in conjunction. Furthermore, combining the optimizations requires only minor code changes. The work demonstrates how to combine our techniques with the widely used CUSP SpMV algorithm and the currently highest performing yaSpMV algorithm to significantly improve both algorithms' performance. This work experimented with a wide range of matrices, and show that the modified version of CUSP on average reduces vector I/O by 37% and reduces the total I/O by 31%, while the modified version of yaSpMV reduces the vector and total I/O by 36% and 31%, resp. proposed work improve CUSP's total throughput by 14% on average and up to 77% for certain matrices, and improve yaSpMV's throughput by 12% on average and 35% for some matrices.

J. Huang, J. Ren, W. Yin and L. Wang,[8] Sparse Matrix-Vector Multiplication (SpMxV) algorithms suffer heavy performance penalties due to irregular memory accesses. In this examination introduce a novel compressed element storage (CES) format, in which the additional data structures for indexing are abandoned, and each location associated with the non-zero element of the matrix is now indicated by the name of a variable multiplied by the corresponding element of the vector. To ensure fastest access and parallel access without data hazards, on-chip registers are used exclusively to replace the BRAM or off-chip DRAM/SRAM to hold all the SpMxV data. On-chip DSP resources are fully utilized so as to ensure a maximum number of multipliers concurrently working.

4. PROBLEM STATEMENT

Given many problems in scientific computing result in large matrices, it is of interest to determine the extent to which this performance can be maintained for such matrices. Although in previous examination only demonstrates the scheme for the case that only one MVM fails, it can be extended to the case that multiple MVMs fail by selecting the codes with larger distance. The previous work can be extended in following ways.

- Hardware architectures for banded matrices and symmetric matrices that can significantly extend the scalability to large order matrices and achieve higher degrees of parallelism,
- Hardware architectures that can trade parallelism with FPGA resources to achieve greater scalability.

5. CONCLUSION

This work reported an extensive survey of literature on an efficient fault-tolerance design for integer parallel matrix–vector multiplications. Various previous works are examined to write review on matrix multiplication. The implementation of sparse matrix sparse vector multiplication on a reconfigurable computing platform provides a unique solution to limitations often encountered in software programming. Though fault avoidance and fault removal approaches such as use of high quality components, better and conservative designs and fabrications practices, and test and validation processes are effective in reducing the faults, they cannot completely eliminate in complex systems. Coding for computers must satisfy very restrictive speed and reliability constraints. Since encoders and decoders can also fail, the fault tolerant implementations are gaining importance.

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.

