A Brief Survey on Efficient Fault-Tolerance Design for Multiplications

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-Now-a-days new generation embedded computers are requested by all designing frameworks, for example, communication, generation, nuclear power plants, flying machine, autos, and so on. The embedded PC frameworks are getting to be mind boggling in both their structure and architecture. An embedded PC framework can be utilized progressively frameworks A survey on plan dependent on error remedy coding has been exhibited to ensure parallel filters. In that plot, each filter is treated as a bit, and excess filters that go about as equality check bits are acquainted with recognize and right errors. In this short, applying coding methods to secure parallel filters is tended to in an increasingly broad manner. Specifically, it is demonstrated that the way that filter sources of info and yields are not bits but rather numbers empowers a progressively efficient insurance. This lessens the security overhead and makes the quantity of excess filters autonomous of the quantity of parallel filters. The present writing audit is portrayed. At last, both the adequacy in securing against errors and the expense are assessed for a field-programmable gate array usage in past work.

Keywords-Fault tolerance, matrix vector multiplication, parallel processing, soft errors.

I. INTRODUCTION

Traditionally, the problem of computational fault-tolerance has been solved through modular redundancy. In this technique, several identical copies of the system operate in parallel using the same data, and their outputs are compared with voter circuitry. If no errors have occurred, all outputs will agree exactly. Otherwise, if an error has occurred, the faulty module can be easily identified and the correct output determined. Modular redundancy is a general technique and can be applied to any computational task. Unfortunately, it does not take advantage of the structure of a problem and requires a large amount of hardware overhead relative to the protection afforded.

A more efficient method of protecting computation is to use an arithmetic code and tailor the redundancy to the specific operation being performed. Arithmetic codes are essentially error-correcting codes whose error detecting and correcting properties are preserved during computation. Arithmetic codes offer performance and redundancy advantages similar to existing error-correcting codes used to protect communication channels. Unfortunately, arithmetic codes exist for only a limited number of operations.

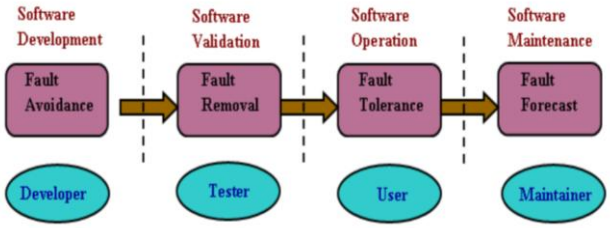

Fig. The Transition of Fault, Error and Failure in a Software Cycle.

This review addresses the general problem of designing an arithmetic code to protect a given computation. A practical arithmetic code must satisfy three requirements:

• It must contain useful redundancy.

• It must be easily encoded and decoded.

• Its inherent redundancy must be preserved throughout computation.

The first two requirements are shared by error-correcting codes for protecting data transmission, while the third requirement is unique to arithmetic codes. This host of requirements makes designing practical arithmetic codes an extremely difficult problem.

A. Fault Tolerance

Fault tolerance is the property that enables a system to continue with its correct operation even in the presence of faults (errors), and it is generally implemented by error detection and subsequent system recovery. Fault tolerance has been a subject of research for a long time, and significant amount of work has been produced over the years to provide fault tolerance, systems are usually designed such that some redundancy is included. The common types of redundancy used are information, hardware, and time redundancy.

Error-detecting and error-correcting codes provide fault tolerance while using information redundancy, i.e. the data includes additional information (check bits) that can verify the correctness of the data before it is used (error-

detection), or even correct erroneous data bits (error-correction). Different error-detecting and error-correcting codes have been proposed including parity codes, cyclic codes, arithmetic codes etc. The major disadvantage of error-detecting and error-correcting codes is that they are limited to errors that occur during transfer of data (system bus) or errors in memory.

Fault-tolerant control systems are closed-loop systems in which a satisfactory performance can be maintained regardless of uncertainty. The sources of uncertainty, for instance, are the occurrence of a component malfunction and exogenous disturbances. Fault-tolerant design problems in dynamic processes have attracted great attention in both theoretical research and industrial applications. In general, there is a trade-off between the fault-tolerance capability and the achievable performance.

Fault-tolerant control systems are complex since they are required to operate despite the presence of uncertainty in the systems. The main purpose is to design a closed-loop system with fault- tolerant controller or observer that takes the component faults into account to ensure a stable closed-loop system with a sufficiently small degradation in the nominal performance. Thus it is by nature a multi-objective design exercise. The most suitable formulation is in the form of linear matrix inequalities (LMIs).

As many control systems and engineering processes become more and more complex, widespread and integrated, the effects of system failures can be simply devastating to the infrastructure of modern society for extensive commercial, industrial and safety reasons. Therefore, fault tolerance is necessary in many control engineering applications such as: safety-critical systems (nuclear reactors, aircraft, and missile guidance systems), cost-critical systems (large space structures, space vehicles, autonomous underwater vehicles).

The terminology used in this thesis is listed below and is consistent with that defined by the committee.

- Fault: An unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable, usual or standard condition.
- Failure: A permanent interruption of a system's ability to perform a required function under specified operating conditions.

Note that the difference between fault and failure is that the term fault is to represent a malfunction in the process of system operation while failure denotes the situation that system functions suffer a complete breakdown. For instance, in the electronic parallel circuit design, a short circuit is always considered as a failure while a open circuit is considered as a fault since the damage of a open circuit to the whole circuit depends on the location of occurrence.

A typical fault-tolerant control system consists of three parts: a reconfigurable controller, fault detection (FD) and diagnosis scheme and a control law reconfiguration mechanism. One of the key challenges, which are the main objective of this thesis, is to design a sufficiently robust controller which is reconfigurable. Note that the controller design methods assume that the fault information is provided by some FD schemes.

In particular, this thesis concentrates on the control design for systems with faults/failures in the actuators and sensors (components) as well as the disturbances from the external environment. Figure 1.1 depicts the considered uncertainty in the system. Note that we define the faults/failures by the location of occurrence.

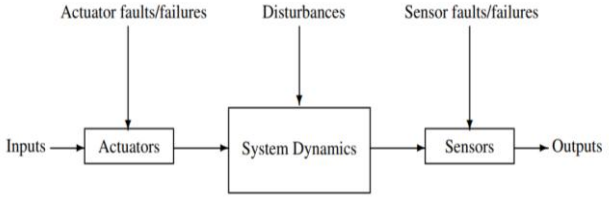

Figure 1.1 Uncertainty in an Open Loop System.

Failures are common in distributed runtimes that operates on thousands of computers, especially when the computation infrastructure is built using commodity hardware equipments. Although this is different from the experience we have in using high end computation clusters with better networking equipments, and also in using leased resources (virtual machines) from Cloud providers, we also identify the need for producing distributed runtimes with fault tolerance capabilities.

Therefore, applications can process input data amidst node failures, provided that the number of replicas of data and the replica placement can effectively handle failures. Further, their approach of writing intermediate data products to persistent storage simplifies the failure handling logic.

II. SYSTEM MODEL

A. Architecture of Error Detection

The architecture shown in Figure 2.1 consists of two processing nodes (processors), a shared memory and a Compare & Control Unit (CCU) connected through a shared bus. In such architecture RRC is performed as follows. Each job is duplicated and concurrently executed on both processing nodes. At a given time (a checkpoint request), the execution of the job is interrupted and a checkpoint is taken at each node. The checkpoint includes sufficient information such that the job can be resumed from that particular point. We consider a checkpoint to be represented as the state (status) of a processing node. Once the states of both nodes are obtained, each processing node

sends its state to the CCU. The CCU compares the states from both processing nodes. If the states match, i.e. no errors are detected; the CCU stores one of the states in memory and signals to the processing nodes to continue with the execution of the job. If the states do not match, i.e. an error is detected; the CCU loads the most recently saved state from memory and sends it to both processing nodes forcing them to roll-back the execution of the job.

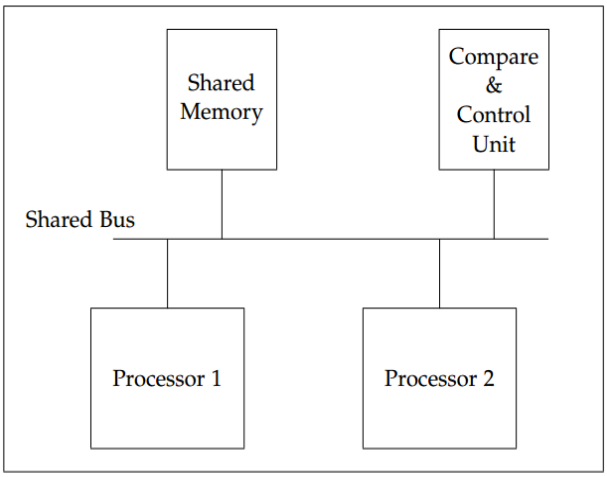

Figure 2.1: Error detection system model.

B. Fault Model and Fault Assumptions

For the fault model, we consider that soft errors (faults) that occur in the processing nodes cause erroneous outcome of the undergoing computation, i.e. bit-flips in the result produced after some computation. The fault model considers the occurrence of soft errors as an independent event. This means that the occurrence of a soft error does not depend on previous soft errors that have occurred. Further, the fault model considers that the probability Pt that no errors occur in a processing node within an interval of length T is given. This model is not limited to the number of faults that can occur within a time interval, which is an assumption that has been used in other research studies.

While soft errors can occur in any part of a computer system, i.e. memories, communication controllers, buses, etc , we address soft errors that occur only in the processing nodes, and we assume that errors occurring elsewhere in the system are handled with conventional techniques for fault tolerance, e.g. error-correction codes (ECC) for handling soft errors that occur in memory. Further, we assume that each soft error provides a unique erroneous outcome. By using this assumption, if two soft errors occur, one in each processing node, the states of both processing nodes will differ due to that each soft error has caused a different erroneous outcome. Figure 2.2 shows the model of fault tolerance parallel filter.

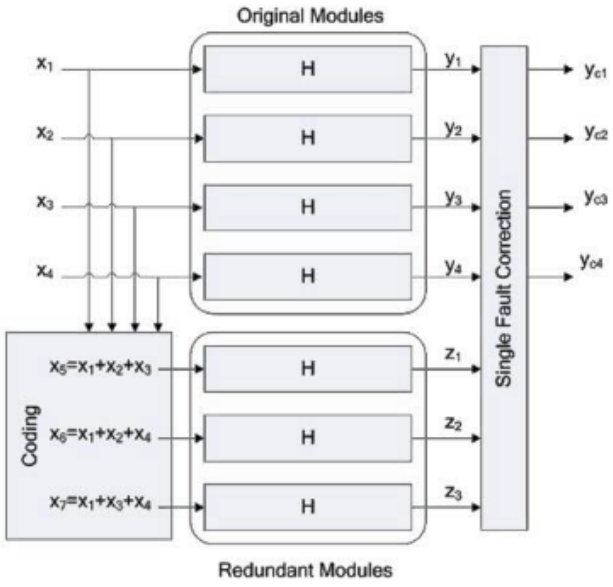

Figure 2.2 Fault Tolerances in Parallel Filter.

III. LITERATURE SURVEY

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	This paper proposes a fault-tolerant design for integer parallel MVMs. The scheme combines ideas from error correction codes with the self-checking capability of MVM.
2	Utilizing Multi-Level Data Fault Tolerance to Converse Energy on Software-Defined Storage	S. Chen et al.	2017	This study enables an energy-efficient multilevel data fault tolerance design on a single disk group. A series of experiments show that the proposed design could reduce the storage system energy consumption significantly when compared with the original architecture.
3	Two-State Check pointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems	M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali and J. Henkel	2016	This paper presents a low-overhead two-state check pointing (TsCp) scheme for fault-tolerant hard real-time systems. It differentiates between the fault-free and faulty execution states and leverages two types of checkpoint intervals for these two different states.

4	Locality-Aware Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication on Many-Core Processors	M. O. Karsavuran, K. Akbudak and C. Aykanat	2016	In this work, we identify five quality criteria for efficient and scalable thread-level parallelization of SpMMTV that utilizes one-dimensional (1D) matrix partitioning. We also propose two locality-aware 1D partitioning methods, which achieve reusing A-matrix non zeros and intermediate z-vector entries.
5	Fast Sparse Matrix and Sparse Vector Multiplication Algorithm on the GPU	C. Yang, Y. Wang and J. D. Owens	2015	We examine the scalability of three approaches -- no sorting, merge sorting, and radix sorting -- in solving this problem. For breadth-first search (BFS), we achieve a 1.26x speedup over state-of-the-art sparse-matrix dense-vector (SpMV) implementations.
6	Research on parallel model for sparse matrix-vector iterative multiplication	J. Li, Q. Wu and P. Zou	2013	we have established three different parallel algorithm models and analyzed the characteristic features of their computing and communication, and through the analysis and comparison of time-cost, we choose the optimal model.
7	Design of Fault Tolerant Reversible Arithmetic Logic Unit in QCA	B. Sen, M. Dutta, D. Banik, D. K. Singh and B. K. Sikdar	2012	this work targets design of reversible ALU (arithmetic logic unit) in QCA (Quantum-dot Cellular Automata) framework. The design is based on the reversible QCA structure (RQCA) introduced in this paper.

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 paper proposes 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.

S. Chen et al. [2] In the era of digitalization, people store their data digitally on local or remote storage. To prevent data loss due to disk failures, data fault tolerance mechanisms have been widely deployed to provide different levels of reliability. However, applying data fault tolerance mechanisms incurs extra cost on storage space and energy consumption. Therefore, data with different importance should be stored on storage that provides different levels of fault tolerance to lower the cost. Software-defined storage (SDS) become a viable option since it includes a centralized controller to process data requirements. However, there is little discussion on how to enable multilevel data fault tolerance on single SDS disk group to satisfy different levels of fault tolerance requirement with energy-efficient considerations. %In addition, applying two or more data fault tolerance mechanisms directly on the same disk group could be problematic and not energy efficient. To address this issue, this study enables an energy-efficient multilevel data fault tolerance design on a single disk group. A series of experiments show that the proposed design could reduce the storage system energy consumption significantly when compared with the original architecture.

M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali and J. Henkel [3] Check pointing with rollback recovery is a well-established technique to tolerate transient faults. However, it incurs significant time and energy overheads, which go wasted in fault-free execution states and may not even be feasible in hard real-time systems. This paper presents a low-overhead two-state check pointing (TsCp) scheme for fault-tolerant hard real-time systems. It differentiates between the fault-free and faulty execution states and leverages two types of checkpoint intervals for these two different states. The first type is non uniform intervals that are used while no fault has occurred. These intervals are determined based on postponing checkpoint insertions in fault-free states, with the aim of decreasing the number of checkpoint insertions. The second type is uniform intervals that are used from the time when the first fault occurs. They are determined so as to minimize execution time for faulty states, leaving more time available for energy management in fault-free states. Experimental evaluation on an embedded processor (LEON3) and an emerging nonvolatile memory technology (ReRAM) illustrates that TsCp significantly reduces the number of checkpoints (62% on average) compared with previous works, while preserving fault tolerance. This results in 14% and 13% reduced execution time and energy consumption, respectively. Furthermore, we combine TsCp with dynamic voltage scaling (DVS)

and achieve up to 26% (21% on average) energy saving compared with the state-of-the-art techniques.

M. O. Karsavuran, K. Akbudak and C. Aykanat [4] Sparse matrix-vector and matrix-transpose-vector multiplication (SpMMTV) repeatedly performed as z←ATxand y← A z (or y A w) for the same sparse matrix A is a kernel operation widely used in various iterative solvers. One important optimization for serial SpMMTV is reusing A-matrix nonzeros, which halves the memory bandwidth requirement. However, thread-level parallelization of SpMMTV that reuses A-matrix nonzeros necessitates concurrent writes to the same output-vector entries. These concurrent writes can be handled in two ways: via atomic updates or thread-local temporary output vectors that will undergo a reduction operation, both of which are not efficient or scalable on processors with many cores and complicated cache-coherency protocols. In this work, we identify five quality criteria for efficient and scalable thread-level parallelization of SpMMTV that utilizes one-dimensional (1D) matrix partitioning. We also propose two locality-aware 1D partitioning methods, which achieve reusing A-matrix nonzeros and intermediate z-vector entries; exploiting locality in accessing x-, y-, and z-vector entries; and reducing the number of concurrent writes to the same output-vector entries. These two methods utilize rowwise and columnwise singly bordered block-diagonal (SB) forms of A. We evaluate the validity of our methods on a wide range of sparse matrices. Experiments on the 60-core cache-coherent Intel Xeon Phi processor show the validity of the identified quality criteria and the validity of the proposed methods in practice. The results also show that the performance improvement from reusing A-matrix nonzeros compensates for the overhead of concurrent writes through the proposed SB-based methods.

C. Yang, Y. Wang and J. D. Owens [5] We 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. We examine the scalability of three approaches -- no sorting, merge sorting, and radix sorting -- in solving this problem. For breadth-first search (BFS), we 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.

J. Li, Q. Wu and P. Zou [6] The most effective algorithms of solving large sparse linear system are Block Wiedemann and Block Lanczos, sparse matrix-vector multiplication iterations is the main process of these algorithms, to achieve parallel computing of its process, we have established three different parallel algorithm models and analyzed the characteristic features of their computing and communication, and through the analysis and comparison of time-cost, we choose the optimal model.

B. Sen, M. Dutta, D. Banik, D. K. Singh and B. K. Sikdar [7] this work targets design of reversible ALU (arithmetic logic unit) in QCA (Quantum-dot Cellular Automata) framework. The design is based on the reversible QCA structure (RQCA) introduced in this paper. A fault tolerant architecture of reversible ALU is also synthesized. The proposed designs are verified and evaluated over the existing ALU designs and found to be more efficient in terms of design complexity and quantum cost.

IV. PROBLEM IDENTIFICATION

In the previous work the efficient coding schemes for fault - tolerant parallel filters are the filters that has been used in the filter bank for the communication channel in the protection status. The data's communicated through this channel length have been arrived from the associated architecture In many cases, the filters perform the same processing on different incoming signals as there is a tendency to use multiple-input-multiple- output systems .This parallel operation can be exploited for fault tolerance. In fact, reliability is a major challenge for electronic systems. In particular, soft errors are an important issue, and many techniques have been proposed over the years to mitigate them. Some of these techniques modify the low-level design and implementation of the integrated circuits to prevent soft errors from occurring. Other techniques work at a higher abstraction level by adding redundancy that can detect and correct errors.

V. CONCLUSION

Technology scaling has empowered us to keep pace with the power, execution, zone and usefulness necessities of electronic circuits. Alongside the focal points, it has additionally given difficulties because of expanded spillage current, unwavering quality disappointments, and so forth. At first delicate errors were a worry just for security basic applications. Yet, with the scaling of innovation, delicate errors are getting to be relevant notwithstanding for electronic gadgets in purchaser showcase space. Because of the restrictive expense related with the structure, producing and different pledges required for coordinated circuits, regularly gadgets planned with solid accentuation in one market section will discover use in another, for example items reused crosswise over inventory (purchaser) and car markets.

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] S. Chen et al., "Utilizing Multi-Level Data Fault Tolerance to Converse Energy on Software-Defined Storage," 2017 IEEE International Conference on Smart Cloud (SmartCloud), New York, NY, 2017, pp. 1-6.

[3] M. Salehi, M. Khavari Tavana, S. Rehman, M. Shafique, A. Ejlali and J. Henkel, "Two-State Checkpointing for Energy-Efficient Fault Tolerance in Hard Real-Time Systems," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 7, pp. 2426-2437, July 2016.

[4] M. O. Karsavuran, K. Akbudak and C. Aykanat, "Locality-Aware Parallel Sparse Matrix-Vector and Matrix-Transpose-Vector Multiplication on Many-Core Processors," in IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 6, pp. 1713-1726, 1 June 2016.

[5] 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.

[6] J. Li, Q. Wu and P. Zou, "Research on parallel model for sparse matrix-vector iterative multiplication," Proceedings of 2013 3rd International Conference on Computer Science and Network Technology, Dalian, 2013, pp. 122-125.

[7] B. Sen, M. Dutta, D. Banik, D. K. Singh and B. K. Sikdar, "Design of Fault Tolerant Reversible Arithmetic Logic Unit in QCA," 2012 International Symposium on Electronic System Design (ISED), Kolkata, 2012, pp. 241-245.