International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

An Efficient FPGA-Based Dynamic Partial Reconfigurable Implementation

Raghunath B H[*1], Dr. Aravind H S[2]

Submitted: 15/10/2022 **Revised**: 22/12/2022 **Accepted**: 03/01/2023

Abstract: Today's system developers can choose from many electronic gadgets. There are simple integrated circuits, programmable microcontrollers, bespoke chips, and more complex logic devices on the market. FPGA technology is famous for rapid prototyping and implementing small-unit systems. They offer high logic density and the ability to readily upgrade established designs to meet new standards or change system function or structure. FPGAs have a shorter design cycle than custom devices and can use low-cost design tools. These benefits reduce FPGA design NRE. Their weakness is radiation [1]. This primarily involves SRAM-based FPGAs, which are in high demand because they have high throughput at a low cost. The design of fault-tolerant systems can reduce the number of errors they experience.

A fault-tolerant FPGA design approach is presented as a proposal in this study. This technique can be utilized in systems with a constrained redundancy area and cannot use excess resources while operational. In order to alleviate some of the issues with the system, we will implement FPGA partial dynamic reconfiguration. This technique's primary focus is recovery from both temporary and permanent flaws. SEU faults will be simulated via errors in the FPGA configuration memory. After the job, an analysis of the solution's hardware overhead and the effectiveness of the secured system design is carried out.

Keywords: *Fault Tolerant System, Generic Controller, Partial Reconfiguration, Triple Modular Redundancy.*

1. Introduction

Manufacturers have reduced processors and transistors. Chip power doubles every two years, per Moore's law. 1965's idea lasted 50 years. When ICs drop to 14 nm [2], the law will be void.

Smaller transistors boost performance, reduce power consumption, and lower prices. Wires and gadgets with fewer atoms and bonds are fragile and more prone to defects. Stress and the environment affect these gadgets. These factors can alter device fabric and performance. Small data nodes are more radiation-prone and use less power.

Recently, related Junocircled Jupiter in 2016. Its radiation belts are stronger [3]. Spacecraft electronics must be radiation- and fault-resistant. Juno uses BAE Systems' RAD750 processor [4]. Missing Juno's orbit is costly and time-consuming.

Dependability matters. Dependable, maintainable, durable, safe, and secure. It is trustworthy. System-reliability mechanisms exist. Fault-tolerant system design survives failure. Faults slow but do not stop operations. Reliability increases with redundancy. Copying and comparing circuits can uncover and disclose system problems.

Electronics gives system engineers options. Textiles include programmable microcontrollers and ICs. Small-unit systems and prototyping employ FPGA. Meet new standards or modify system function or structure easily. FPGAs are faster and cheaper than custom devices. FPGAs reduce NRE. [5] They are radiation-vulnerable. This uses cheap SRAM-based FPGAs. Fault-tolerant system design reduces failures.

Reconfigurable systems increase fail-safety. Reconfigure FPGA. DPR allows fault localization and mitigation. This reuses FPGA. Custom circuit designs can avoid destroying FPGA resources and keep programs running. This enhances system uptime.

A fault-tolerant FPGA design is proposed. This solution requires no extra space or resources. Dynamic FPGA reconfiguration decreases system issues. They are targeting temporary and permanent flaws. FPGA mistakes imitate SEUs. Hardware and architecture analysis ends the work.

2. Goals of The Research

Various fault-tolerant system applications require different levels of reliability. The breadth of hardware on which a bad design can be executed is crucial. This is especially true for long-term missions when hardware must be reliable (in terms of years).

The following is a synopsis of the objectives of the study, which may be found in the research:

(1) To offer a way for designing a digital system in FPGA that can recover from transient and persistent faults.

   i. The system's planned architecture operates in a constrained implementation region; thus, it can only use FPGA resources assigned at the start of its life.

1 Assistant Professor, Acharya Institute of Technology, Bangalore, Karnataka – 560 107, India
2 Professor, JSS Academy of Technical Education, Bangalore, Karnataka – 560 060, India
** Corresponding Author Email: raghunathbh.publicatons@gmail.com*

ii. FPGA modules are unaffected by a transient fault in one system module.
iii. If a system's architecture must be changed to recover from a permanent fault, the new one must continue to produce accurate outputs and be fault-tolerant.

(2) To create the controller for reconfiguration that controls FPGA fault mitigation after PDR. It provides defect detection, localization, type, and reconfiguration information. It can induce synchronization as needed.
(3) Create a test platform to evaluate the suggested methodology's methodologies and procedures. Fault injection tests the survivability of FT architectures created using methodological principles.

3. Methodology

This paper describes the fundamentals of a suggested technique for protecting systems by implementing their elements as fault-tolerant systems in FPGA. The method uses precompiled setups to mitigate faults.

LIA refers to collecting FPGA resources given to implement some dependability-critical system parts. This implementation region is specified during system design and cannot be changed. During permanent fault recovery, this assessment limits fault mitigation. The limited implementation area statement makes it possible to build deterministic fault mitigation scenarios, simplifying fault reduction for both temporary and permanent problems based on previously prepared parameters. Offline optimization reduces the hardware and performance overhead of fault mitigation strategies.

3.1. Methodology - Basic Principles

The solution safeguards digital FPGA systems. It is a formula for changing FPGA system architecture for fault recovery to increase its life. Long-term missions, where the small implementation area with every design error, justify such tactics: combined fault detection, localization, and mitigation.

FT architectures are picked from the original system's several portions. Comparing repeated functional units in FT and CED designs facilitates identification and localization. Unplanned. Mitigation needs PRM-level localization. When a PRM fails, its fault model must be determined. Transient vs. permanent fault mitigation exists. Both are controlled by GPDR (GPDRC). This unit regulates ICAP interface reconfiguration and fault mitigation. 3.2 gives details.

The approach can detect and correct temporary faults induced by SEU in the FPGA configuration memory, which can lead to faulty system functionality if unchecked. A relocatable golden copy of the PRM unit-type bitstream is needed to mitigate transient defects. When an FT architectural unit causes an issue, the GPDRC reconfigures the PRM. After this procedure and any unit synchronization, the system will usually work.

The methodology also fixes irreversible faults in FPGA physical resources (CLBs, connectivity resources, etc.). Permanent fault mitigation depends on many FT design sequences that perform the same function. Functional units and fault-tolerant components are implemented with single PRMs. If the situation is irreparable, download another FT design into the FPGA. It has less diagnostic circuitry. Faulty FPGA components will not be used after FT. A "degradation approach" selects FT designs that implement one system unit in varying numbers of PRMs and excludes the remaining PRM.

3.1.1. Design of the System Methodology

This method results in the production of a new system protected by FT architecture, which guarantees resilience against transitory failures and various persistent flaws that interfere with the proper operation of the FT system.

The following information needs to be specified by the designer on the input side of the securing process:

i. The language used to describe hardware is used to explain the system's design.
ii. The specification of the FPGA is desired.
iii. The limitations imposed by the users on the implementation.
iv. The process of allocating implementation space in the FPGA for a particular system.

The following items are produced as a result of the process of securing:

i. The hardware language describes the Secured system's FT system design, including all the essential components.
ii. The comprehensive configuration bitstream for the initialization of the FPGA's configuration.
iii. The process for fault mitigation employs the collection of bitstreams with a limited configuration.

3.1.2. Fundamentals Underlying the Structure of The Fault-Tolerant System

In the method that has been developed, the design is guarded by FT architecture, which guarantees the method's resilience against transient and permanent faults that interfere with the FT system's ability to function correctly. The technique recommends deploying PRMs. A set of PRMs is what we refer to as a configuration. Each component of the FT system is placed in the same PRM.

Figure 1 displays the approach's FPGA FT architecture. Dynamic FT and static GPDRC are included. GPDRC detects and localizes flaws using FT CED logic units. PRM error signals reach GPDRC (PRR1-PRR4). Splitting FT into several PRMs eliminates PRMs with chronic errors. Signals between modules and a module and the FPGA are routed through a single PRM near all other PRRs.

FT PRMs can allocate the other 4 PRRs. PRRs vary in number. PRR (3,4) can only implement simple FT designs (for example, TMR with the simple voter, duplex with checker). After irreversible defects, he has little opportunity of recovering the system. Fewer alternatives exist. More PRRs reduce errors but increase configuration options.

Several FT architectures illustrated securing a system by degrading particular parts. First, TMR architecture with doubled voter detects voter mistakes. TMR uses unprotected voter units. A duplex with a comparator is the final architecture. This architecture lacks fault tolerance since it cannot tell which two repeated outputs are wrong. This system works until the comparison unit identifies the first error.

3.1.3. Possible Configurations for The FT Architecture

Precompiled FT settings are used for persistent faults. Generation 1 configurations use the same FT and PRM. Figure 2 shows different FT generations.

Every generation has underutilized PRMs. Flags show if the PRM assigned the matching PRR. This system part's generation 0 configuration is 1111. After finding the first permanent fault, the next-generation configuration without the problematic PRM is implemented. When a new problem impacts another PRM, this is done. Incorrect PRMs increase setups. Bitstream relocation reduces memory utilization by preventing duplicate PRM units from being created. Every PRM type requires a single bitstream, except PRM ROUTE. Each setup only stores the PRM ROUTE bitstream.

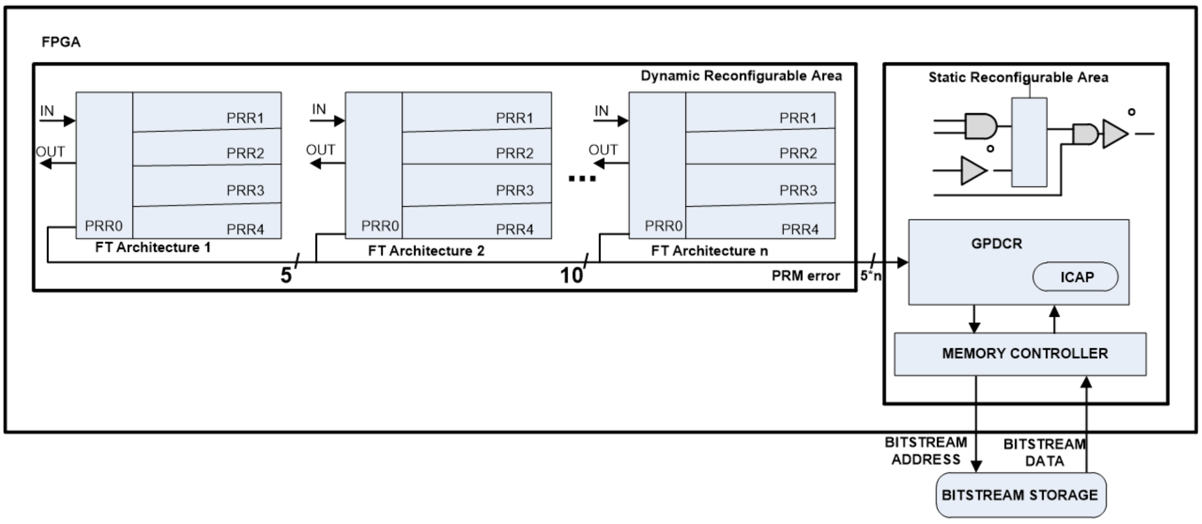

Fig. 1. Fundamental Framework of Proposed Research Approach.

3.1.4. The Bitstream Relocation Method to Reduce the Number of Configurations

Due to Xilinx's design and implementation processes, the generated PRM partial configuration bitstream cannot be allocated to a different PRR. Each PRR with a PRM must have a PRB. If N separate PRMs are to be applied to M PRRs, N * M PRBs must be constructed and kept in external memory for partial run-time Reconfiguration.

Bitstream relocation reduces PRBs to N. These PRBs can be used to reconfigure all PRRs meeting relocation parameters. Design and implementation apply these requirements. This approach requires equivalent FPGA resources for all PRRs.

This approach always generates PRBs for all PRM kinds in one PRR location. Before run-time Reconfiguration, bitstream manipulation is needed to adjust its position information for other PRRs.

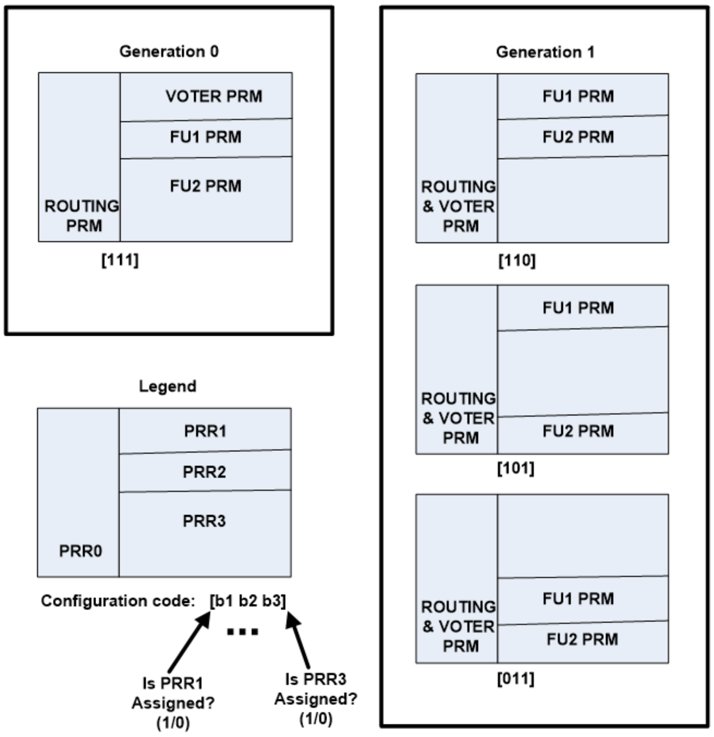

Fig. 2. Many Generations of FT Architectures and Various Ways to Configure.

3.1.5. Synchronization Issues

Reconfiguring a failing PRM in a replicated FT design can produce synchronization concerns. During Reconfiguration, other units run with individual states. This problem occurs because PRM's PDR uses sequential circuits during transient fault mitigation. Another synchronization issue may occur when the FT architecture is reconfigured, and all PRMs are changed to prevent permanent faults. Its active components are reset, and any input or output system portions are paused. PDR fault mitigation raised synchronization concerns.

Several synchronization solutions were suggested to recover the TMR unit. Different target systems. [6] describes a checkpoint method for consecutive FSMs. The reconfigured unit must be readily reachable while awaiting other units. [7] discusses softcore CPUs. After Reconfiguration, any units can be put into stop mode, and their memory registers written. Other synchronization methods are used in packet processing. Until the next packet arrives, hide the modified unit's output. [8] describes how to prevent defective outputs from the reconfigured unit from causing additional Reconfiguration by temporarily disabling the system's fault detection unit for a given minimum (the most extended amount of time that can pass between the arrival of two consecutive packets). Local reset synchronizes units while this window is open.

This breakthrough explored synchronizing massive sequential circuits by replicating another repetitive unit's state. The synchronization-capable architecture's voting unit has flaw detection, localization, and a simple control mechanism. Figure 3 depicts TMR architecture.

Due to the synchronization approach, each unit is connected to the ring by oriented point-to-point (source-destination) links. These links face the ring. When the enable signal is off, functional units reveal state register values. Enabled units save previous units' values in their state registers. Once the values of all registers are recorded, the unit sends the sync signal to the voter. The voter hides an unsynchronized unit's fault signals until the sync termination impulse. This prevents the Reconfiguration of an unsynchronized unit in an indeterminate state.

This synchronization method is available for usage in FT architectures whenever it is required to do so. It is possible to create the implementation for FT systems based on duplexes in a manner that is analogous to TMR. The detection logic of PRM ROUTE can govern state copying in place of a voter unit.

The amount of overhead caused by this method is proportional to the number and size of the FU state registers. This procedure was explained in [9].

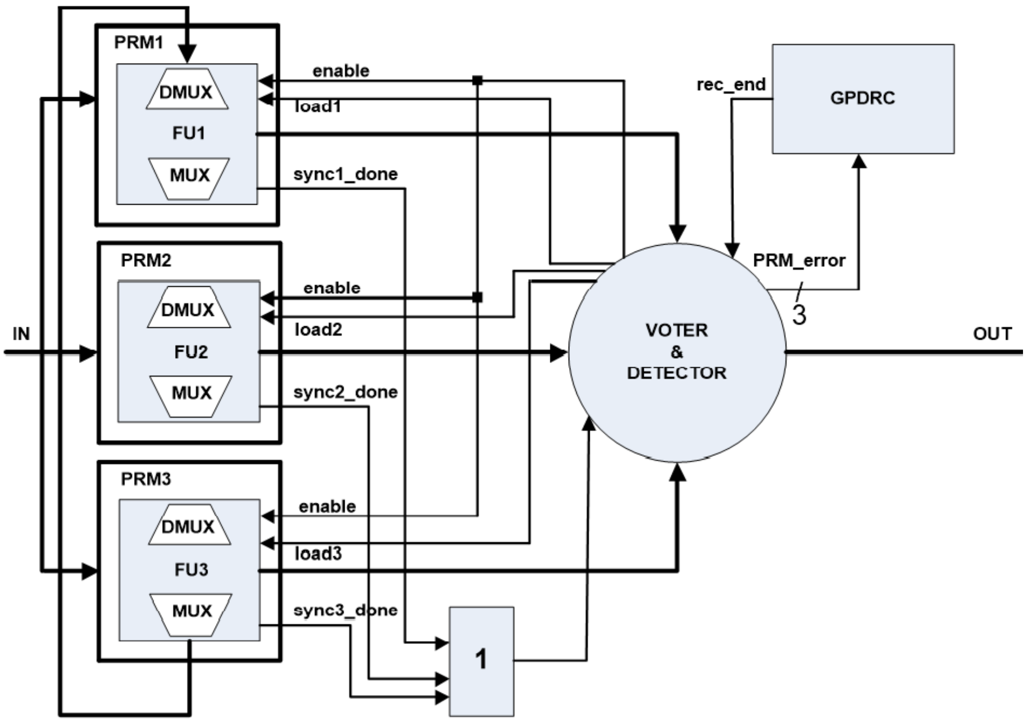

Fig. 3. FT Architecture with Unit Synchronization [9].

3.2. Generic Partial Dynamic Reconfiguration Controller

Microprocessors control dynamic reconfigurations. Reconfiguration's universal processors may waste power. Wasted performance adds power, complexity, and failure risk. Software errors can delay or disrupt microprocessor reconfiguration. The general PDR reconfiguration controller was designed as hardware to reduce resource utilization and failure possibility.

The first GPDRC for fault mitigation appeared in [10]. [11] demonstrated the implementation using a counter and SEU injection. When a PRM fails permanently, the GPDRC can reconfigure the entire FT system, which comprises numerous PRMs. The new controller moves loaded PRBs, chooses the next-generation configuration, and synchronizes the FT system. [12] proposed the GPDRC to reduce transient and permanent faults.

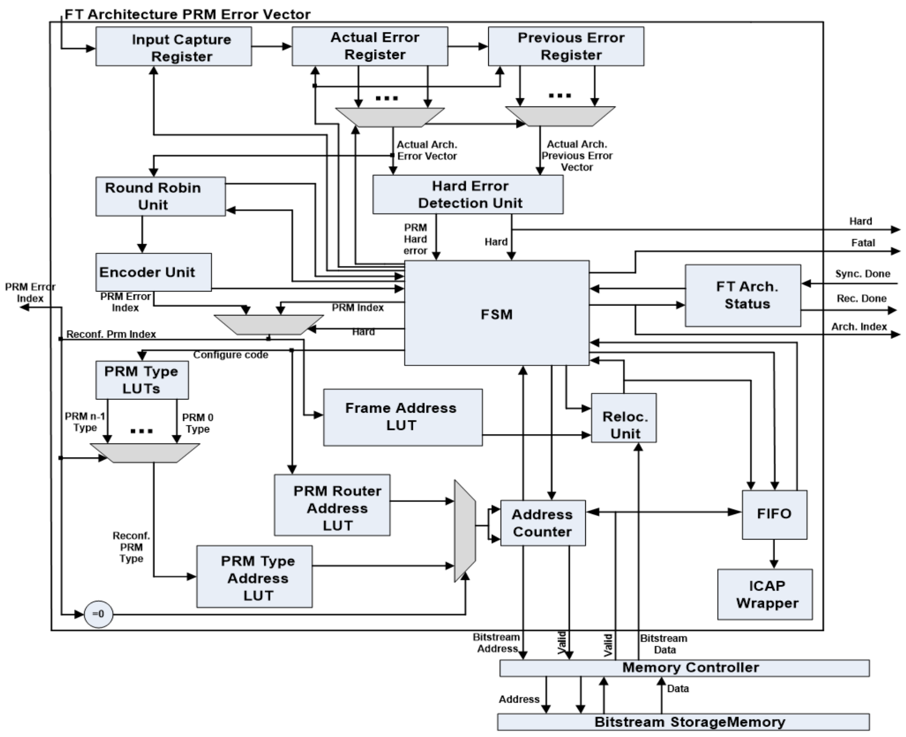

Fig. 4. Structure of Fault Resistant System using SRAM for FPGA [12].

3.2.1. Objectives of the GPDRC's Design

Prior to the construction of the GPDRC, several design goals that needed to be accomplished were identified as follows:

- New controllers with universal softcore processors must use fewer resources.
- In settings with variable PRM numbers, the controller must be generic. The controller can and should be smaller if the system is simple and has few PRMs.
- The number of subsequent reconfiguration cycles should help the controller identify if a PRM issue is transitory or permanent (false PRM detection, PRM reconfiguration, PRM synchronization). If the fault occurs fewer than n times, it is transitory.
- PDR will be done at maximum speed utilizing ICAP on Xilinx FPGAs (up to 100MHz).
- To reduce precompiled PRBs, the controller must utilize the same PRB for numerous PRMs' PDRs.
- Before synchronization, the controller should disregard the changed PRM error output. FT can synchronize autonomously or with an external controller.
- The controller should support external bitstream memory. Bitstream data transmission must be universal for compatibility with external memory controllers.

3.2.2. Design of GPDRC Unit

Figure 4 depicts GPDRC's architecture. The interface accepts FT error vectors. This component's width depends on FT architectures and PRMs. FPGA's ICAP interface and external bitstream storage communicate bitstream address, data, and validity indicators. Sync done and rec done regulate FT PRM synchronization.

The PRM error index vector shows the PRM that GPDRC maintains, whereas the arch. index vector shows the current FT architecture for fault mitigation. Strong signal suggests persistent PRM. The fatal signal means GPDRC cannot build FT since insufficient PRMs.

The GPDRC has multiplexers, LUTs, nine primary units, and FIFO (MUX). Input register errors are stored in the error register while a GPDRC reconfiguration cycle runs over FT error signals. The error vector from the previous cycle is stored to determine the problem. Complex error detection may designate a flaw as permanent if it is identified twice.

The round-robin unit checks for transient faults in the register and delivers its index to the encoder unit if none are found. It picks PRM units. Each configuration's PRM index and type are stored in the LUT. Address counter uses bitstream's memory address after PRM type resolution. Index = 0 uses the PRM bitstream routing address.

Permanent failures cause FT architecture PRM reconfiguration. PRM loops from 0 to PRM - 1. FT fault signals create LUT configuration codes. Mistake PRM's routing error 0 signal does not affect FT architecture. Resolving an indexed PRM bitstream address uses the same methods.

The address counter addresses each bitstream data word. Since only one copy of each unit type is stored, bitstream must be relocated. The relocation unit adjusts the bitstream frame address using the updated PRM's Frame Address LUT.

The FT Architecture Status unit's rec done signal is set when all problematic PRMs are PDR'd. GPDRC ignores architecture problems until PRMs are externally synchronized. After activating sync-done, GPDRC will not ignore FT failure signals. By excluding them from GPDRC, our approach enables multiple FT synchronization techniques.

3.2.3. Fault-Tolerant Implementation of GPDRC Unit

A defense mechanism against SEUs should be developed in the GPDRC's reconfigurable architecture. Radiation-resistant cloth is one choice. Also, FT. In this scenario, the GPDRC must be relocated inside the active FPGA. The GPDRC's error output must also be connected to the FPGA. GPDRC's FT design mitigates errors. This technique comes before any PRM fault mitigation. Because only one ICAP instance can be accessed, this approach is not included in GPDRC instances.

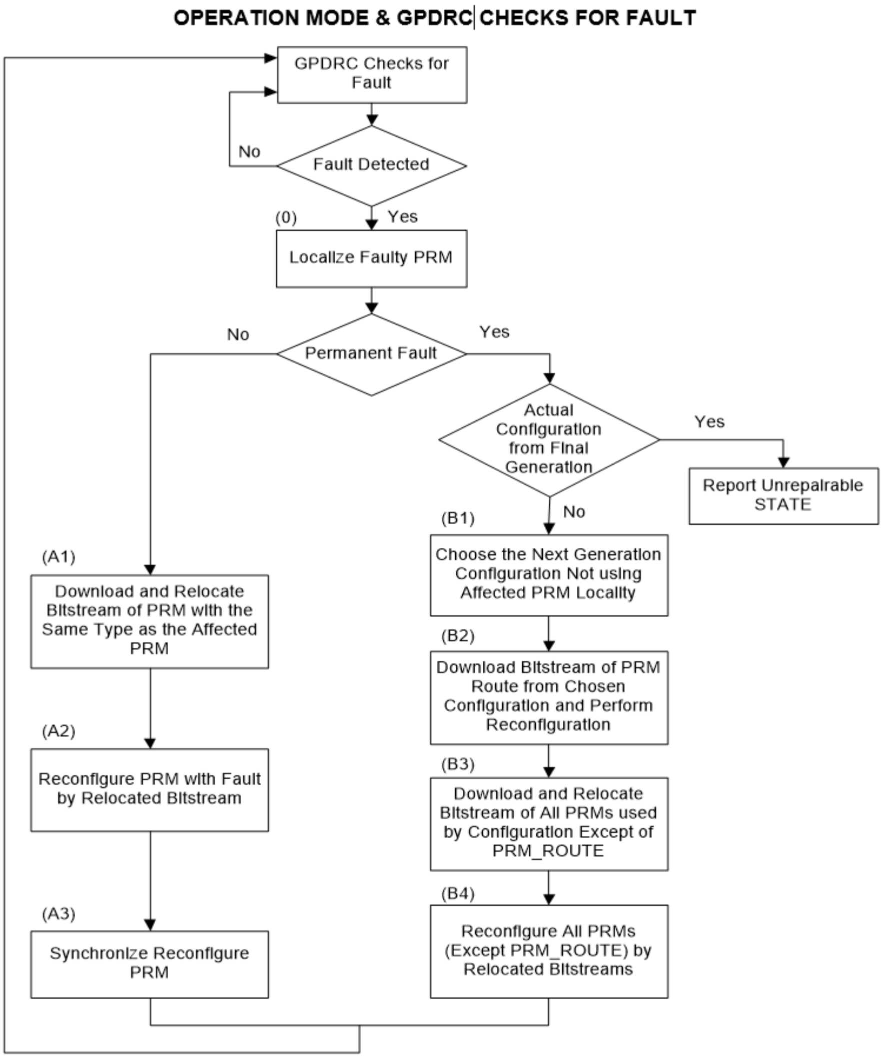
OPERATION MODE & GPDRC CHECKS FOR FAULT
GPDRC Checks for Fault
Fault Detected
No
Yes
(0)
Localize Faulty PRM
Permanent Fault
No
Yes
Actual Configuration from Final Generation
Yes
No
Report Unrepairable STATE
(A1)
Download and Relocate Bitstream of PRM with the Same Type as the Affected PRM
(A2)
Reconfigure PRM with Fault by Relocated Bitstream
(A3)
Synchronize Reconfigure PRM
(B1)
Choose the Next Generation Configuration Not using Affected PRM Locality
(B2)
Download Bitstream of PRM Route from Chosen Configuration and Perform Reconfiguration
(B3)
Download and Relocate Bitstream of All PRMs used by Configuration Except of PRM_ROUTE
(B4)
Reconfigure All PRMs (Except PRM_ROUTE) by Relocated Bitstreams

Fig. 5. Flow Diagram of Reconfiguration.

3.3. Fault Mitigation Procedure

Figure 5 depicts system operation following PRM fault detection. FT error signals indicate a faulty PRM (step 0). Functional units and voters are built using independent PRMs, and their relationship is understood.

The GPDRC evaluates if the erroneous PRM is temporary or permanent when localized. Option A is for temporary defects. The following steps depend on whether the issue can never be corrected, independent of configuration creation. GPDRC stores the last generation's configuration code. If this is the last generation, this new permanent fault cannot be prevented, and the FT architecture will tell the GPDRC. Need outside help (e.g., the physical placement of configuration is moved to another locality of FPGA, or the FPGA is replaced with a new one). Option B works when the present setup is not complete.

Case 1 - Rehabilitation from a problem that was just temporary

GPDRC from external memory reads the PRB corresponding to the erroneous PRM. The GPRDC has an understanding of the unit's configuration as well as its kind and PRM distribution. The downloaded PRB is sent to the most appropriate PRR, as in Figure 6 (step A1).

The next step in mitigating the problem is to reroute this bitstream such that it can be used to modify the problematic PRM. GPDRC is the driving force for the PRM reconfiguration and the relocated PRB (step A2).

After Reconfiguration, the PRM must be synced with other FT components. GPDRC controls synchronization (step A3).

Case 2 - Recovering from a permanent error

When PRM detects a permanent failure and the current configuration is not from the last generation, a new configuration from the next generation is chosen. The malfunctioning PRM will not be used. The GPDRC chooses a configuration based on code that responds to the bitwise negation of FT fault signals (B1 step).

For example, say the current configuration is 111 (generation 0), and a voter unit defect is permanent. The FT error signal vector on GPDRC will be 100. This value's bitwise negative is the

generation 1 configuration code 011.
The external bitstream storage stores the PRB for PRM ROUTE. This bitstream reconfigures PRR0 (the only FPGA PRR where it can be allocated). This means this PRB is not moving (step B2).
Next, download all remaining PRB copies implementing PRMs. Configuration determines the number of bitstream copies needed and their type. As with transient fault reconfiguration, all PRBs are downloaded from the exact location. Each downloaded PRB will be relocated to fit applicable PRRs (step B3).
The downloaded and moved PRBs are utilized to reconfigure PRMs (step B4). After Reconfiguration, PRMs are locally reset. This stage can also involve synchronization (state recovery of all impacted PRMs).

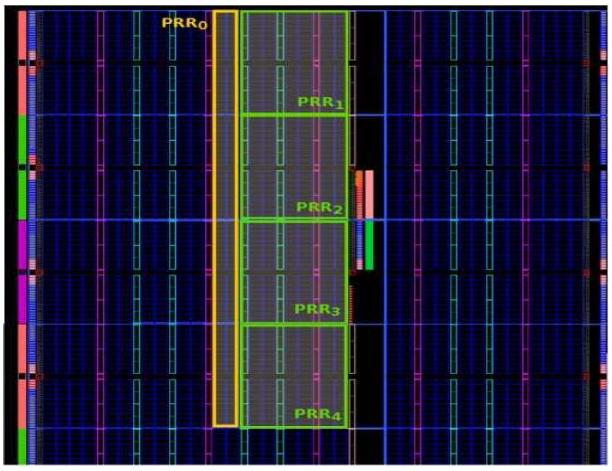

Fig. 6. Dislocation of Partial Reconfiguration Zones within FPGA belongs to FT Architecture.

4. Experimental Results

This section discusses the technique's design and execution results. It shows implementation details and additional hardware overhead for TMR-secured designs. Analyze GPDRC installation outcomes on secured systems. A transient fault simulator tests a platform's capacity to recognize, locate, and mitigate transient faults. Permanent flaws in the created system are simulated, and it is verified that problematic modules may be avoided by modifying FT architecture.
All experimental systems were designed using Xilinx 14.7. The target FPGA was Xilinx's Virtex 7 XC7S100.

4.1. Implementation of GPDRC

GPDRC helps design secure systems. It is a smaller, speedier alternative to softcore controllers. The system's size depends on how many PRMs it has employed.

4.1.1. Implementation of GPDRC on a Generic Level and Scaling

Partitioned systems can use GPDRC. It is a generic unit with definable attributes for varied designs.
Several FT system parts. This option controls PRB storage. PRM COUNT shows how many PRMs each FT architecture has to implement units. The product of these generic values is the width of GPDRC and many system PRM error vectors. GPDRC handles FT and PRM architectures. Save only one copy of each PRM type to reduce defects. Type index width determines the PRM type. PRM TYPE WIDTH is PRM's square root. Finally, Address WIDTH.
Counters, registers, decoders, and other logic evaluated GPDRC resource utilization for several system partitioning methods. GPDRC size is unaffected by system complexity. Several variables affect it, including PRMs. FPGA design included FT and PRM designs. Testing 3 to 6 PRMs. FT Architecture started employing TMR with the duplex voter when each design required five or six PRMs. All four PRMs started with TMR, voter, and duplex with the checker.
Table 1 compares MicroBlaze's PDR IP core with GPDRC's modules. 32 GPDRC-controlled FT designs with 6 PRMs were analyzed. Columns include unit name (column 1), unit size in slices (column 2), occupied LUTs and FlipFlops (columns 3 and 4), and TMR alternative size (5).

Table 1. The quantities of FPGA resources dedicated to the GPDRC (32 FT architectures, 6 PRM per FT architecture).

XC7S100 Virtex 7	*Size*	*LUTs*	*F/Fs*	*TMR*
Input Capture Register	38 (0.5%)	73	168	98 (2.4X)
Actual Error Register	37 (0.5%)	78	78	102 (2.4X)
Previous Error Register	37 (0.5%)	164	168	104 (2.4X)
Hard Error Unit	2 (0.1%)	3	0	7 (2.8X)
Round Robin Unit	4 (0.1%)	4	4	12 (2.7X)
Error Encoder	2 (0.1%)	2	0	5 (1.8X)
Relocation Unit	4 (0.1%)	12	1	17 (2.6X)
Architecture Status Unit	2 (0.1%)	36	24	5 (2.8X)
Address Counter	14 (0.2%)	38	15	42 (2.2X)
FSM	14 (0.2%)	36	12	46 (2.3X)
Others (LUTs, MUXs...)	112 (1.3%)	264	154	378 (2.7X)
GPDRC total	266 (3.7%)	710	624	816 (2.0X)
MicroBlaze	628 (7.7%)	1414	1491	1664 (2.8X)

4.1.2. Amount of Time Needed for PRM to Reconfigure

Controllers must measure reconfiguration time. Table 2 shows GPDRC metrics by PRM bitstream size. Reconfiguration time is determined by PRB size, not PRM resource use.
The table contains multiples of the Virtex 7 FPGA's smallest PRM, which has 20 CLBs. Multiples of the smallest PRM in column 1 are mentioned in columns 2 through 3 and 4.

Table 2. Amount of Time Required to Reconfigure one PRM.

XC7S100 Virtex 7 CLBs in multiples of the size of the smallest PRM	*CLBs [#]*	*Bitstream Length [kB]*	*Reconfiguration Time [ms]*
1X	20	5	0.23
2X	40	10	0.42
3X	60	15	0.67
4X	80	20	0.93
5 X	100	25	1.14

4.2. Evaluation of Hardware Overhead

This section describes the FT architectures developed for each generation (0 and 1), along with their qualities and restrictions. These architectures are models for describing system methodology. Different FT architectures can identify and localize PRM errors.
The suggested FT designs use 5 PRMs; hence five error signals are emitted from the PRM ROUTE block. These signals indicate

a malfunction at GPDRC's inputs.

4.2.1. Fault Tolerant Architecture of Generation 0

Initial Generation 0 FT architecture is built on TMR, where the majority element checks all FU outputs (voter). 3 PRM FUs, 1 PRM VOTER, and 1 PRM ROUTE make up this architecture. Figure 7 shows this architecture's structure.

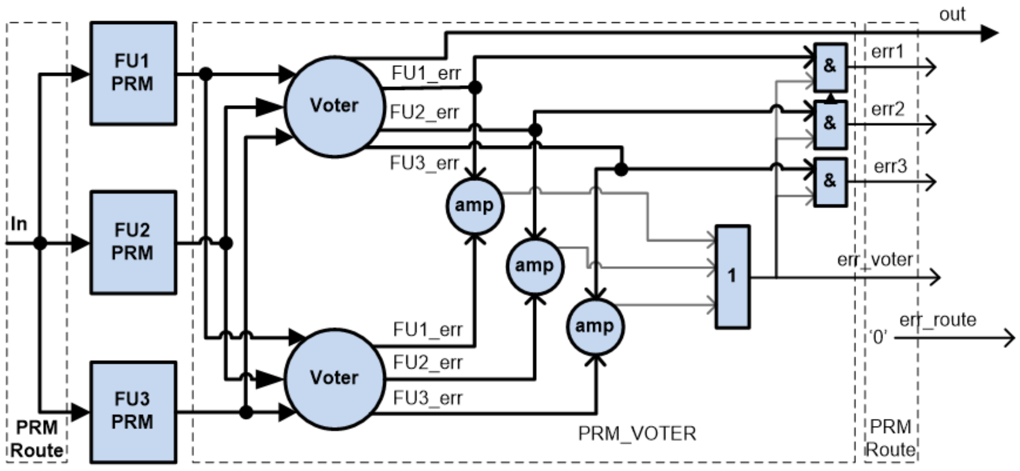

Fig. 7. Generation 0 of FT Architecture based on TMR.

Each FU has its PRM and no diagnostic logic. All PRM FU outputs connect to PRM VOTER. PRM VOTER features a voter and diagnostic logic for FU fault detection (comparators, logic gates). The voter's duplex design detects problems and modifies the PRM VOTER block. Voter design can also use two-rail logic. GPDRC reconfigures the PRM based on PRM VOTER error outputs, then selects the appropriate bitstream from bitstream storage. PRM ROUTE is the FT PRM-FPGA interface. Because this architecture's PRM ROUTE block lacks diagnostic logic or FPGA logic elements, it is not protected against failures, and the error signal err route is always logic zero.

The architecture may fail if most aspects of a duplex design cannot be identified. Small logic mitigates this hazard. FT outputs may indicate inaccurate values while adjusting PRM VOTER and PRM ROUTE.

4.2.2. Architecture Fault Tolerant of Generation 1

The architecture of the first generation of FTs is a duplex that includes a PRM CHECKER unit. The structure of Generation 1 FT can be shown in Figure 8. This architecture consists of four PRMs (2 PRM FU, PRM CHECKER, and PRM ROUTE). Each FU is realized as a single PRM, and the error-controlled output multiplexor is responsible for switching between the PRM's outputs. Checker is a PRM that is played solo.

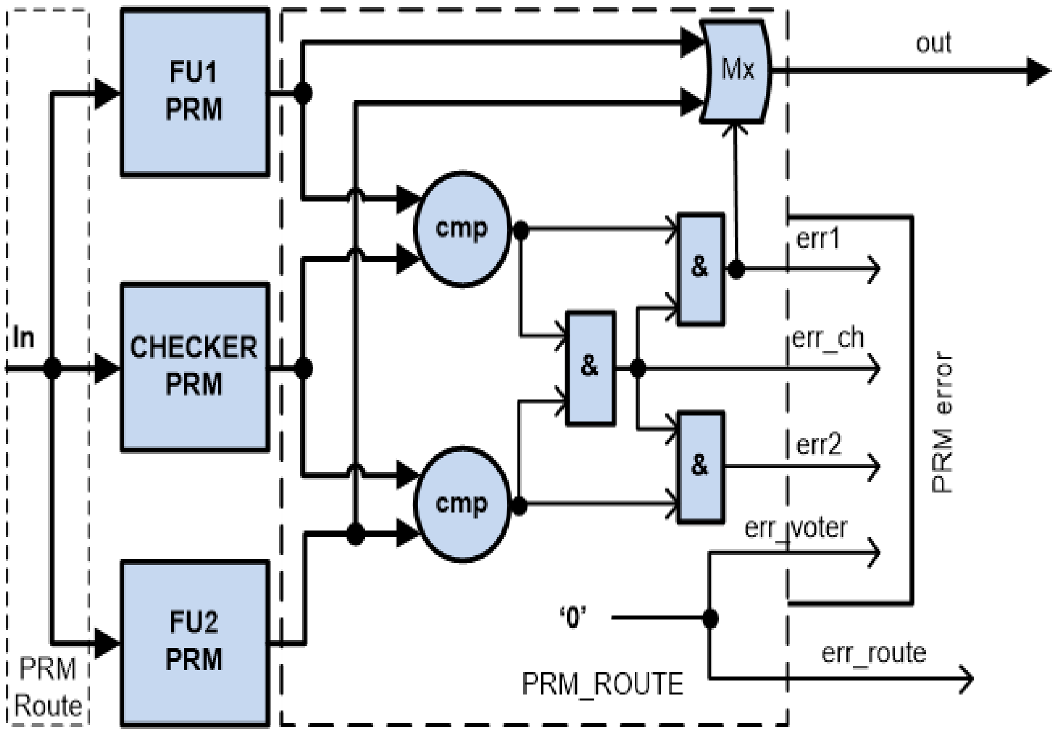

Fig. 8. Generation 1 of FT Architecture based on Duplex with Checker Protocol.

4.2.3. Evaluation of Resource Overhead

Table 3 shows FT components that generate FPGA hardware overhead. This table shows our methods' unit overhead. PRM ROUTE and PRM VOTER are generation 0 overheads. Any three FU sizes were not considered overhead because they were in the standard TMR architecture. PRM VOTER unit size was lowered without using incorrect unit localization to acquire only the overhead for our solution. For the same reasons as generation 0, generation 1 overhead contains PRM ROUTE units solely. Column 1 shows FU output width in bits; column 2 shows generation 0 overhead in slices; column 3 shows generation 1 overhead in slices.

Table 3. The overheads of Generations in Slices.

XC7S100 data width [bits]	*Generation 0 [slices]*	*Generation 1 [slices]*
2	10	4
4	18	9
8	30	13
16	54	24
32	98	42
64	168	78

5. Conclusion

This section outlines the process of securing a system design to reduce fault occurrences and fault mitigation steps.

System designer designed FT-based subsections. CED is required for fault detection, and logic pinpoints incorrect PRM. GPDRC drives fault mitigation. This controller picks the scenario based on fault type and previous issues.

PDR reduces transitory faults, and FT ensures correct outputs. Moving the PRB golden copy corresponding to the appropriate PRM type reduces the stored precompiled PRBs. When a persistent flaw is detected, and an alternative FPGA configuration is available, the FT architecture is changed. This method enables many configurations. They are characterized by their ability to handle erroneous PRMs. GPDRC can pick which system configuration to employ if an issue persists. FT's hardware overhead was compared to TMR's. One FT architecture requires less hardware. Fewer PRMs make the GPDRC a lower priority. More considerable PRBs slow Reconfiguration. Overhead and PRB size must be balanced (and the reconfiguration time).

In the future, Markov models will be built for the proposed fault recovery FT structures. Moreover, Pipelined microprocessor partitioning experimented with hardware overhead and PRB size.

References

[1] M. Wirthlin, E. Johnson, N. Rollins, M. Caffrey, and P. Graham. The reliability of FPGA circuit designs in the presence of radiation-induced configuration upsets. In Field-Programmable Custom Computing Machines, 2003. FCCM 2003. 11th Annual IEEE Symposium on, pages 133-142, 2003.

[2] Flynet E., Yuffie M. and Knoll E. 4.1 14nm 6th-generation core processor soc with low power consumption and improved performance. In 2016 IEEE International Solid-State Circuits Conference, ISSCC 2016, San Francisco, CA, USA, January 31 - February 4, 2016, pages 72-73, 2016.

[3] Patrick Blau. Juno: Spacecraft information. URL: <http://spaceflight101.com/juno/spacecraft-information/>. Accessed: 2016-07-06.

[4] John Rhea. Bae systems move into third-generation rad-hard processors. Military & Aerospace Electronics, 13(5), 2002.

[5] Aiwu Ruan, Bairui Jie, Li Wan, Junhao Yang, Chuanyin Xiang, Zujian Zhu, and Yu Wang. A bitstream readback-based automatic functional test and diagnosis method for Xilinx FPGAs. Microelectronics Reliability, 54(8):1627-1635, 2014.

[6] Conrado Pilotto, Jose Rodrigo Azambuja, and Fernanda Lima Kastensmidt. They synchronize triple modular redundant designs in dynamic partial reconfiguration applications. In SBCCI '08: Proceedings of the 21st annual symposium on Integrated circuits and system design, pages 199-204, New York, NY, USA, 2008. ACM.

[7] Yoshihiro Ichinomiya, Shiro Tanoue, Motoki Amagasaki, Masahiro Iida, Morihiro Kuga, and Toshinori Sueyoshi. Improving the robustness of a softcore processor against us by using tmr and partial Reconfiguration. In Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM '10, pages 47-54, Washington, DC, USA, 2010. IEEE Computer Society.

[8] Jason A. Cheatham, John M. Emmert, and Stan Baumgart. A survey of fault-tolerant methodologies for FPGAs. ACM Trans. Des. Autom. Electron. Syst., 11(2):501-533, 2006.

[9] Miculka L. and Kotasek Z. Synchronization technique for tmr system after dynamic Reconfiguration on FPGA. In The Second Workshop on Manufacturable and Dependable Multicore Architectures at Nanoscale (MEDIAN 2013), pages 53-56. Politecnico di Milano, 2013.

[10] M. Straka, J. Kastil, and Z. Kotasek. Generic partial dynamic reconfiguration controller for fault-tolerant designs based on FPGA. In NORCHIP '10, pages 1-4, Washington, DC, USA, 2010. IEEE CS.

[11] Straka M., Miculka L., Kastil J. and Kotasek Z. Test platform for fault tolerant systems design qualities verification. In 15th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems pages 336-341. IEEE Computer Society, 2012.

[12] Miculka L. and Kotasek Z. Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into FPGA. In 17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, pages 171-174. IEEE Computer Society, 2014.

Mr. Raghunath is working as an Assistant Professor at AIT, Bengaluru, and He is a Research Scholar at JSSATE, Bengaluru, affiliated with Visveswaraya Technological University. He has completed B E in electronics and communication engineering from Mysore University and M.Tech degree from Visveswaraya Technological University. His research interest includes Fault tolerance in FPGA, VLSI design, and Embedded System design.

Dr. Aravind H S is working as Professor at JSSATE, Bangalore. He obtained his B E degree in Electronics and Communication Engineering from Bangalore University and M.Tech from the University of Mysore. He obtained his Ph.D. from Visveswaraya Technological University, Belagavi. His research interest includes signal processing, Fault Tolerance computing, and instrumentation.