IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 4, 2022, accepted April 17, 2022, date of publication April 22, 2022, date of current version April 29, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3169495

A Low-Overhead Reconfigurable RISC-V Quad-Core Processor Architecture for Fault-Tolerant Applications

SATYAM SHUKLA, (Student Member, IEEE), AND KAILASH CHANDRA RAY, (Member, IEEE)
Department of Electrical Engineering, Indian Institute of Technology Patna, Dayalpur Daulatpur, Patna, Bihar 801106, India

Corresponding author: Satyam Shukla (satyam.pee17@iitp.ac.in)

This work was supported in part by the Ministry of Electronics and Information Technology, Government of India, New Delhi, through the Project "Special Manpower Development Program for Chips to System Design," under Project R&D/SP/EE/DEIT/SMD/2015-16/126; and in part by the Visvesvaraya Ph.D. Scheme under Grant MeitY/PhD/2582.

ABSTRACT Radiation can affect the correct behavior of an electronic device. Hence, the microprocessors used for space missions need to be protected against fault. TMR (Triple modular redundancy) is used for mitigating various kinds of faults in an electronic circuit. Although TMR provides an excellent level of reliability, it takes a large area and suffers from high power consumption. To reduce the area and power overheads DMR (double modular redundancy) is used. The DMR approach significantly reduces the resource overhead but it also reduces the performance by imposing timing penalty. Various methods have been proposed since the incarnation of TMR and DMR, but the resource overhead is still a challenging issue. Hence, in this work, a new DMR based reconfigurable quad-core RV32IM processor architecture is proposed for fault-tolerant applications. Depending upon the environment of operation and application sensitivity, the designed processor can be reconfigured to work either in normal mode or fault-tolerant mode. The novelty of the proposed architecture is that the reconfigurable feature reduces the resource overhead and makes the processor energy-efficient by optimally using all four processor cores to provide fault-tolerant results. The proposed computing architecture is designed using Verilog HDL(Hardware Description Language) and synthesized on 32nm CMOS (Complementary Metal-Oxide Semiconductor) process technology node using Synopsys Design Compiler EDA (Electronic Design Automation) tool. Compared to unprotected design, the synthesis tool reports -21.75% reduction in power with a time penalty of +9.96% and area overhead of +17.89% for the proposed fault-tolerant approach. Compared to the state-of-the-art fault-tolerant computing system, the proposed design achieves -2.26% low area overhead with its reliability intact as DMR. Further, the proposed processor is prototyped and tested on FPGA (field-programmable gate array) with fault-injection using SEM (soft error mitigation) IP core.

INDEX TERMS Reconfigurable VLSI architectures, fault-tolerant processor, FPGA implementation, RISC-V ISA, single event upset.

I. INTRODUCTION

Electronic devices used for space applications suffer from damage due to cosmic rays. Technology scaling improves the speed, power, and area of electronic devices. Nevertheless, technology scaling has negative impacts on system reliability [1], [2]. In a harsh radiation environment, the collision of a high-energy particle to a CMOS device may produce a bit-flip in a cell, known as single event upset (SEU). Such radiation impact is more prominent in the case of deeply scaled SRAM-based FPGAs [3]–[5]. Other than the harsh radiation environment, a fault may also occur as a result of hardware aging. This single bit change in an electronic circuit, caused by SEUs or hardware aging, may lead to complete system failure [6]–[8].

The faults are commonly categorized as either permanent or transient. As the name suggests permanent faults are those that can not be recovered over time. The transient faults are short-lived and their effect goes away after some time. The ratio of transient-to-permanent faults is 100:1 or even

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

higher [9]. Hence protection against transient faults has now emerged as a critically important design constraint. Protection against faults is achieved by introducing redundancy, either in space or time. The space redundancy uses separate copies of hardware and the result is generated by majority voting. Hence, space redundancy requires a large area and power. In time redundancy, the task is re-evaluated on the same hardware. Hence, time redundancy takes less area compared to space redundancy but it reduces the performance of the system [10], [11].

The electronic systems [12] should be capable of mitigating faults. The triple modular redundancy (TMR) is a long-time-used approach for mitigating various kinds of faults. TMR uses three copies of a circuit and hence provides an excellent level of reliability, but it requires a large area and suffers from high power consumption [13]–[15]. To reduce the area and power overheads, some approaches use double modular redundancy (DMR) [16]. The DMR approach significantly reduces the resource overhead but it also reduces the performance of the system by imposing timing penalty. To improve the performance with low-overhead, selective TMR approach can be used. The selective TMR approach provides the fault-tolerance capability to frequently used blocks such as arithmetic logic units (ALUs) [17], register files [14], even only multipliers [18]. However, the fault tolerance capability in the selective TMR approach is less compared to the conventional TMR approach [19].

So, there is a trade-off between reliability and resource utilization by the adopted technique of fault mitigation. In [20], the authors have considered reliability and power consumption as two main objectives in designing real-time embedded systems. Their scheme postpones the execution of tasks on spare cores. By doing this it gives an opportunity to suspend the execution of second copy of a task when the execution of first copy of the task completes successfully. So, it leads to reduction in power consumption. The authors report 47.6% reduction in peak power compared to the other schemes. In [21] the authors have proposed a scheme in which execution time overlap between primary tasks and their corresponding replicas are reduced. Since the occurrence of fault is rare, the tasks are commonly completed earlier. So, in case of early or successful completion of the primary task the remaining part of replicas are canceled to save power. Since the introduction of the TMR approach, several protection techniques have been appeared using redundancy in hardware or time, but the resource overhead is still a challenging issue [22]. Therefore, it is necessary to adopt a fault-tolerance technique that can provide low resource overheads while still maintaining the performance and reliability level. For such low-overhead implementation of fault-tolerance multicore platforms provide a great opportunity [23]–[25].

Hence, a new reconfigurable DMR based fault-tolerant quad-core processor architecture is proposed using RISC-V instruction set architecture (ISA). This work is based on the idea that a processor need not be in fault-tolerant mode all the time. Depending upon the environment of operation and application sensitivity, the proposed processor can be reconfigured to work in normal mode or fault-tolerant mode. The novel contribution and implementation details of the proposed work are summarized as follows:

- A new low-overhead fault-tolerant quad-core processor is proposed in this paper. Due to its reconfigurable feature, the proposed processor takes low-overhead to provide fault-tolerant results.
- The proposed processor is implemented using Verilog HDL and further synthesized using 32nm CMOS process technology node.
- The proposed processor is prototyped using a commercially available NEXYS4 DDR FPGA board.
- The resource utilization results are compared with state-of-the-art fault-tolerant processors.
- Soft error mitigation (SEM) IP core of Xilinx Inc. is used for fault injection campaign to evaluate the fault tolerance capability of the proposed processor in laboratory experimental setup.

The rest of the paper is organized as follows; Section II highlights the features of RISC-V ISA and existing fault-tolerant computing systems. Section III presents the proposed processor architecture and its functional operation in both normal mode and fault-tolerant mode. Section IV provides the simulation and implementation results and their comparison with existing state-of-the-art processor architectures. Further, chip-level evaluation (on FPGA) with fault-injection using SEM IP core is presented in this section. Finally, section V concludes the paper.

TABLE 1. Comparisons of open source ISAs [26].

Features	SPARC V8	OpenRISC	RISC-V
Base+Ext			✓
Compact code			✓
Quad FP	✓		✓
32-bit	✓	✓	✓
64-bit		✓	✓
128-bit			✓

II. RISC-V ISA AND EXISTING FAULT-TOLERANT COMPUTING SYSTEMS

The design of a processor starts with choosing an ISA, and one of the primary reasons for using RISC-V ISA is that it is open-source, which makes it free ISA. The RISC-V foundation encourages both open-source and proprietary implementations of the RISC-V ISA specification. This is interesting because this type of license allows designers around the globe to come together and build a strong community chain for further developments. This reduced the complexity and IP issues found in previous research projects based on MIPS [27], SPARC [28], and x86 [29]. Table-1 [26] presents the comparison of open-source ISAs. The Base+Ext (modular approach) feature is only supported by RISC-V ISA. The RISC-V ISA supports 32, 64, and 128-bit implementations, it also provides quad floating-point support. RISC-V ISA appears to be the best candidate out of

existing open-source ISAs due to its modular feature and ease of implementation. The features of RISC-V ISA, its implementations, and related work are presented in this section.

A. FEATURES OF RISC-V ISA

The RISC-V ISA was developed for supporting computer architecture research and education, and now it has become a standard open-source architecture for industry implementations. The RISC-V ISA can be implemented as a base integer ISA (necessary for any RISC-V-based implementation) plus optional standard and non-standard extensions to the base ISA. The base ISA consists of a minimum set of instructions sufficient to provide a reasonable target for processor architecture, compiler, assembler, linker, and operating system [30]. There are two primary base integer variants, RV32I and RV64I, which provide 32-bit or 64-bit user-level address spaces.

RISC-V has been designed to support extensive customization and specialization. The modular approach allows computer architects to implement only those set of instructions that are needed for a target application. It makes the design energy-efficient by cutting-off unnecessary hardware and makes it suitable for domain-specific designs. For RISC-V-based processors [31], the base integer ISA can be extended with one or more optional instruction set extensions, but the base integer instructions can not be redefined [30].

B. EXISTING FAULT-TOLERANT COMPUTING SYSTEMS

A RISC-V ISA-based fault-tolerant microprocessor architecture, derived from a non-fault-tolerant SHAKTI-C Class processor [32] named SHAKTI-F is presented in [33]. This work focus on protecting the design by combining both space and time redundancy. In [34], authors have presented a fault-tolerant ALU in which functional units are customized according to the dynamic profiling of the application. Most executed assembler instructions of a program are identified by the profiling. Various TMR implementations of ALU have been presented as part of a case study for the two most used instructions based on benchmark programs. The architecture of ALU is modified to replicate only those instructions which are most used. The quicksort, matrix multiplication, and Fibonacci series generation are used as a benchmark program to identify the most used instructions.

In [35], the authors have discussed the energy consumption of conventional TMR and 2-stage TMR. In view of the deficiencies in the conventional TMR approach, a novel approach named R-TMR (Reactive TMR) is proposed. The R-TMR is an energy-efficient TMR-based approach that can tolerate both transient and permanent faults for hard-real-time systems. To evaluate the effectiveness of fault-tolerance for the design, a simulation model framework is presented. The reported energy consumption of this design is 48% in case of transient faults and 49% for permanent faults compared to the conventional TMR approach while keeping the reliability intact. In [36] authors have proposed a novel scheme for SEU-tolerant turbo decoders which takes an overhead of 2.2 times than unprotected one.

The impact of radiation on soft-processor implemented using SRAM-based FPGA is presented in [37]. This work presents fault-injection in the RISC-V-based processor, and the design shows more than 90% efficiency against SEU. In [22], a new architecture of a fault-tolerant reconfigurable system is implemented on SRAM-based FPGA, with an integrated soft-core processor. The design is implemented on Xilinx Virtex-5 FPGA, which uses a fault-tolerant configuration engine built using PicoBlaze core [38] for SEU mitigation.

All the existing fault-tolerant approaches address the trade-off between reliability and resource utilization by providing a fixed solution that requires more overheads. To the best of the authors' knowledge, there is no such reconfigurable approach used for achieving fault-tolerance in electronic devices.

Algorithm 1 Instruction Scheduling and Operation of Proposed Fault-Tolerant Quad-Core Processor

```
Initialization;
while enable ≠ 0 do
    if mode_sel = 0 then
        Core1 executes Instr_core1;
        Core2 executes Instr_core2;
        Core3 executes Instr_core3;
        Core4 executes Instr_core4;
    else if mode_sel = 1 then
        Core1 executes Instr_core1;
        Core2 executes Instr_core2;
        Core3 executes Instr_core1;
        Core4 executes Instr_core2;
Level1:
        if Out_Port1 = Out_Port3 then
            Instr_core1 result correct;
            Error_T1 = 0;
        else if Out_Port1 ≠ Out_Port3 then
            error in core1 or core3;
            Re-run Instr_core1 on core1 and core3;
            Error_T1 = 1;
            Go to Level1;
        end if
Level2:
        if Out_Port2 = Out_Port4 then
            Instr_core2 result correct;
            Error_T2 = 0;
        else if Out_Port2 ≠ Out_Port4 then
            error in core2 or core4;
            Re-run Instr_core2 on core2 and core4;
            Error_T2 = 1;
            Go to Level2;
        end if
    end if
end while
```

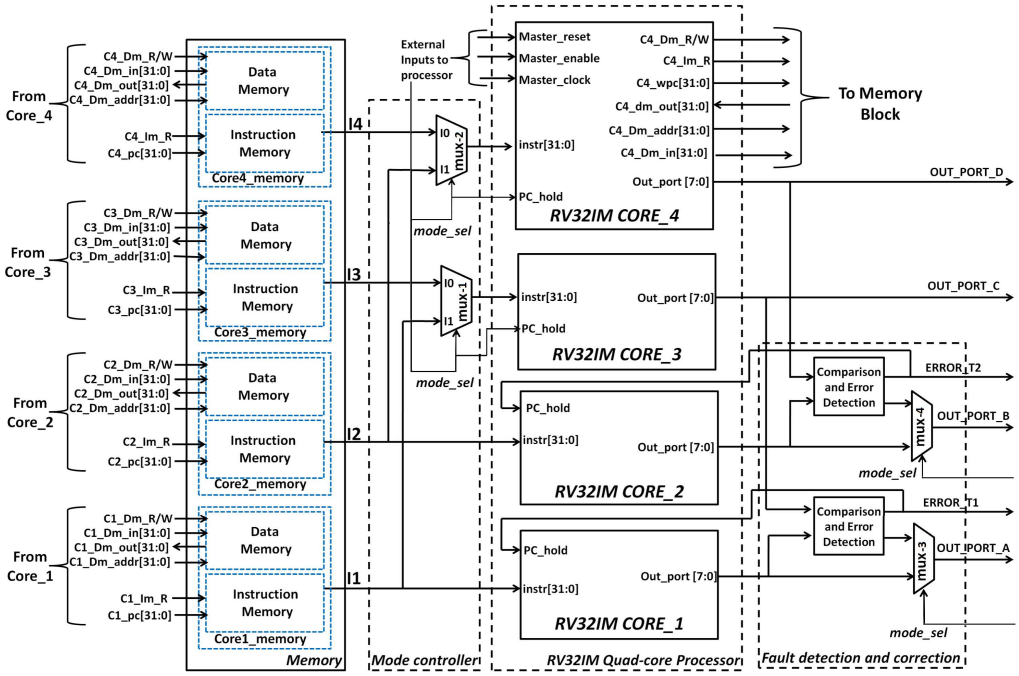

FIGURE 1. Proposed new reconfigurable fault-tolerant quad-core processor architecture.

The proposed processor can be reconfigured to work in normal mode or fault-tolerant mode. Due to the reconfigurable feature, the proposed processor optimally use all four processor cores to provide fault-tolerant result. The reconfigurable approach reduces the use of excessive redundant copies of hardware for achieving fault-tolerant results. Although the area and power overheads are significantly reduced by the proposed approach but this approach also reduces the number of instructions computed per unit time in fault-tolerant mode.

III. PROPOSED RISC-V BASED FAULT-TOLERANT PROCESSOR ARCHITECTURE

The aim is to develop a low-overhead reconfigurable quad-core processor for fault-tolerant applications. RISC-V instruction set architecture appears to be best suitable to design such processor as it is open-source and follows modular approach. Hence, the proposed processor is designed based on RISC-V ISA, and for this work, a single-cycle, 32-bit base integer variant of RISC-V ISA along with standard "M" extension is implemented and will be referred as "RV32IM". The algorithm and architecture of the proposed fault-tolerant, reconfigurable quad-core processor, are presented in this section.

A. NEW FAULT-TOLERANT PROCESSOR ARCHITECTURE

The architecture of a new, low-overhead, DMR based fault-tolerant quad-core processor is presented in figure1. Depending upon the operating environment and application sensitivity, the proposed processor can be reconfigured to work either in normal mode or fault-tolerant mode. The algorithm for instruction scheduling and operation of the proposed processor is presented in algorithm1. The reconfigurable feature allows the design to use its hardware optimally for providing fault-tolerant results. The mode select signal ('*mode_sel*') is used to reconfigure the design. As presented in algorithm-1, when the '*mode_sel*' signal is '0', the processor runs in normal mode of operation, and all four RV32IM processor cores run four different instructions. Hence, the designed processor runs as a basic quad-core processor that can execute four different instructions at a time. In another case, when the *mode_sel* signal is '1', the processor runs in the fault-tolerant mode of operation, and two copies of core1_instruction and core2_instruction are executed. In the fault-tolerant mode of operation, core1 and core3 execute core1_instruction, and core2 and core4 execute core2_instruction. The final result of core1_instruction in fault-tolerant mode is generated after comparing the results of core1 and core3 for every instruction. similarly, The final result of core2_instruction in 'fault-tolerant mode' is generated after comparing the results of core2 and core4 for every instruction. As presented in algorithm-1, if the comparison results are not matched, then an error signal is generated, from the corresponding 'fault detection and correction block'. When an error signal is detected, it guides the architecture to hold the program counter and re-run the previous instruction until the result of both processor cores matches. It is worth mentioning that the result mismatch may be due to a one-bit error or two-bit error. In either case, the processor re-executes the instruction to provide final result.

Continuing to algorithm-1, the proposed reconfigurable, fault-tolerant quad-core processor architecture is presented in

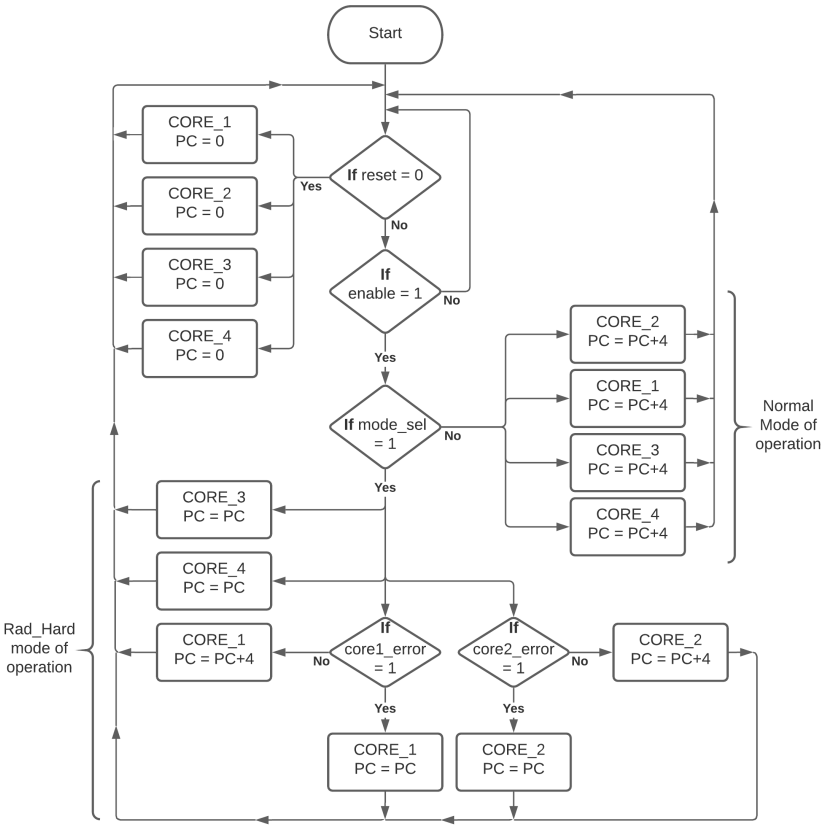

FIGURE 2. Flowchart for content modification of Program counter of proposed fault-tolerant architecture.

figure-1. The proposed architecture consists of four RV32IM cores, a mode controller unit, and a fault detection and correction unit. All four RV32IM cores shown in figure-1 are identical and they access the same memory. Due to space constraint the address, data, and control signals to access memory is shown only for RV32IM_core_4. However, the instruction and data memory segment for each RV32IM processor core along with the signals corresponding to them are shown in the memory block of figure1. The mode controller block takes the '*mode_sel*' signal as an input and based on this signal, the mode of operation for the proposed processor is determined. When '*mode_sel*' signal is '0', mux1 of 'mode controller block' send core3_instruction to core3 and mux2 of 'mode controller block' send core4_instruction to core4 for execution. Hence, the proposed reconfigurable processor works as a basic quad-core processor and executes four instructions simultaneously. In the fault-tolerant mode of operation, the '*mode_sel*' signal is '1', so mux1 sends a copy of core1_instruction to core3, and mux2 sends a copy of core2_instruction to core4. So, two copies of every instruction are executed in the fault-tolerant mode of operation.

In the normal mode of operation, '*mode_sel*' is 0; hence, mux3 of fault detection and correction block shown in figure1 passes the outputs of core1 directly to Port_A, and mux4 of passes the outputs of core2 directly to Port_B. In the fault-tolerant mode of operation, '*mode_sel* = 1', hence, the output of comparison and error detection units are sent to Port_A and Port_B. An error signal is generated by comparison and error detection block in case of result mismatch. The resulting mismatch indicates the occurrence of fault, and hence error signal is generated by the respective comparison and detection unit. In case of the occurrence of an error, the instruction is re-executed. For re-execution of the instruction, the program counter (PC) of the corresponding core is kept on hold until the error signal is present. When the processor operated in fault-tolerant mode PC of core3 and core4 are kept on hold.

In the proposed processor architecture, PC content for each core is of utmost importance as it plays an important role during instruction scheduling. Figure2 shows the flowchart of PC modification with each positive edge of the clock. While reset, all the PC content of all cores is set to zero. As the memory is byte addressable and instruction is of 32-bit word size, in the normal mode of operation PC content of all cores is added with a constant value 4. When the '*mode_sel*' signal is '1', the mode controller block holds the PC content of core_3 and core_4. In fault-tolerant mode, if the error signal is zero, then 4 is added to the PC content of the corresponding core. When an error signal is present, the PC content for the corresponding core is kept on hold, as shown in figure2.

The program counter of the RV32IM core contains the address of the next instruction to be executed. So, when the

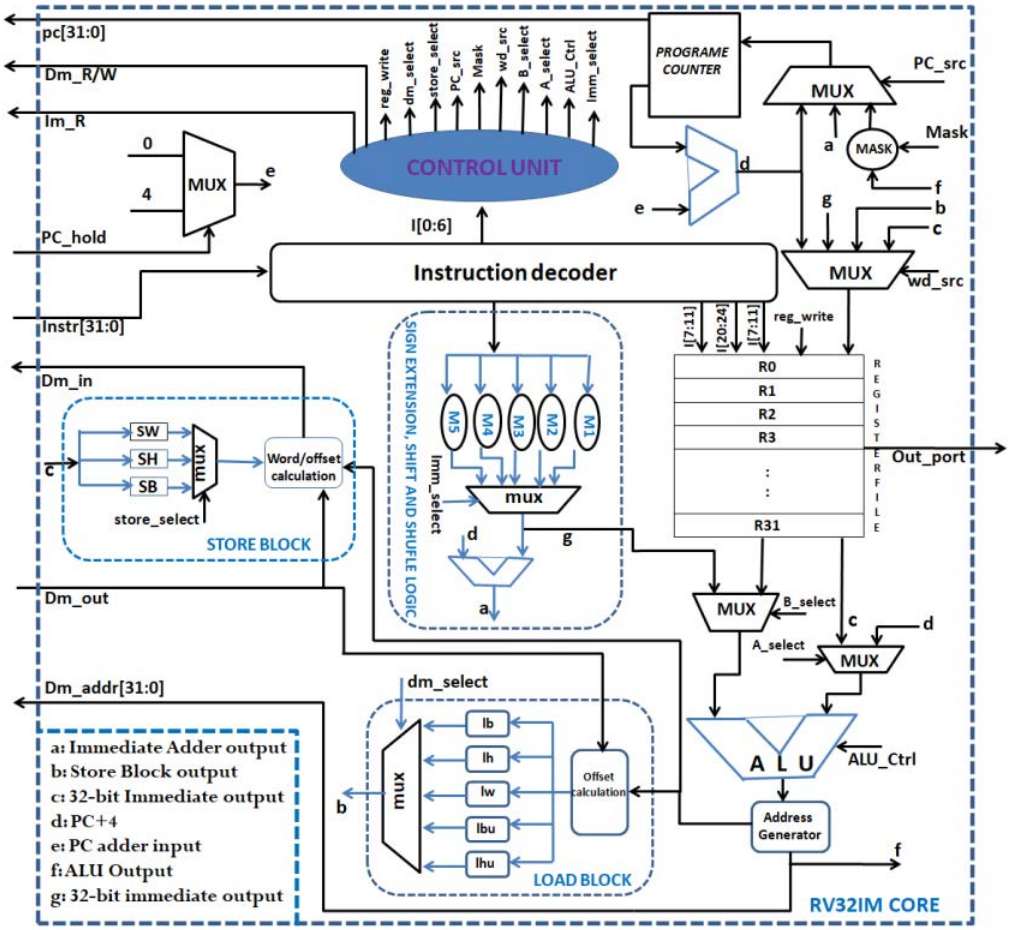

FIGURE 3. Proposed Architecture of single-cycle RISC-V 32IM processor core.

TABLE 2. Comparison of resource overhead of proposed architecture with SHAKTI-F [33].

Evaluation Parameters	SHAKTI-F Architecture [32]					Proposed Processor		
	Base	Fault-tolerant		Improved Fault-tolerant		Base	Fault-tolerant	Overhead
	Resources	Resources	Overhead %	Resources	Overhead %	Resources	Resources	%
Total Instance Count	25176	31766	+26.17	30096	+19.54	44570	52491	+15.09
Std. cell Area (mm^2)	0.13	0.2	+53.80	0.2	+53.80	0.109	0.162	+32.78
Core Area (mm^2)	0.2704	0.3249	+20.15	0.3249	+20.15	0.183	0.223	+17.89
Clock Period (nS)	2.4	3.7	+54.17	3	+25.0	21.78	24.19	+9.96
Total Power (mW)	21.55	10.35	-51.97	11.74	-45.52	5.225	4.2917	-21.75

program counter of an RV32IM core is on hold, it keeps the record of the next instruction to be executed. So, when the processor returns from fault-tolerant mode to the normal mode the processor core resumed the execution from where it left. In the proposed fault-tolerant processor architecture, the RV32IM processor core is one of the main building blocks. Hence, the detailed architecture of the designed RV32IM processor core is presented in the following sub-section.

B. RV32IM PROCESSOR CORE ARCHITECTURE

The RV32IM processor core used in figure-1 is a single-cycle, 32-bit base integer variant of RISC-V ISA along with standard ‘‘M’’ extension. The base integer variant RV32I contains integer computational instructions, integer loads, integer stores, and control-flow instructions. The standard ’M’ extension supports signed and unsigned multiplication and division instructions. The RV32IM is load-store architecture, where only load and store instructions access memory. In load-store architecture, arithmetic instructions operate only on CPU registers. The detailed architecture of the proposed RV32IM core is shown in figure3.

The instructions coming from instruction memory are decoded by the instruction decoder. Based on the op-code of the instructions, the control unit generates appropriate control signals. Since memory is byte-addressable, in every clock cycle the program counter is incremented by 4 to access the next instruction present in the instruction memory. The sign extension and shuffle block is used to handle the

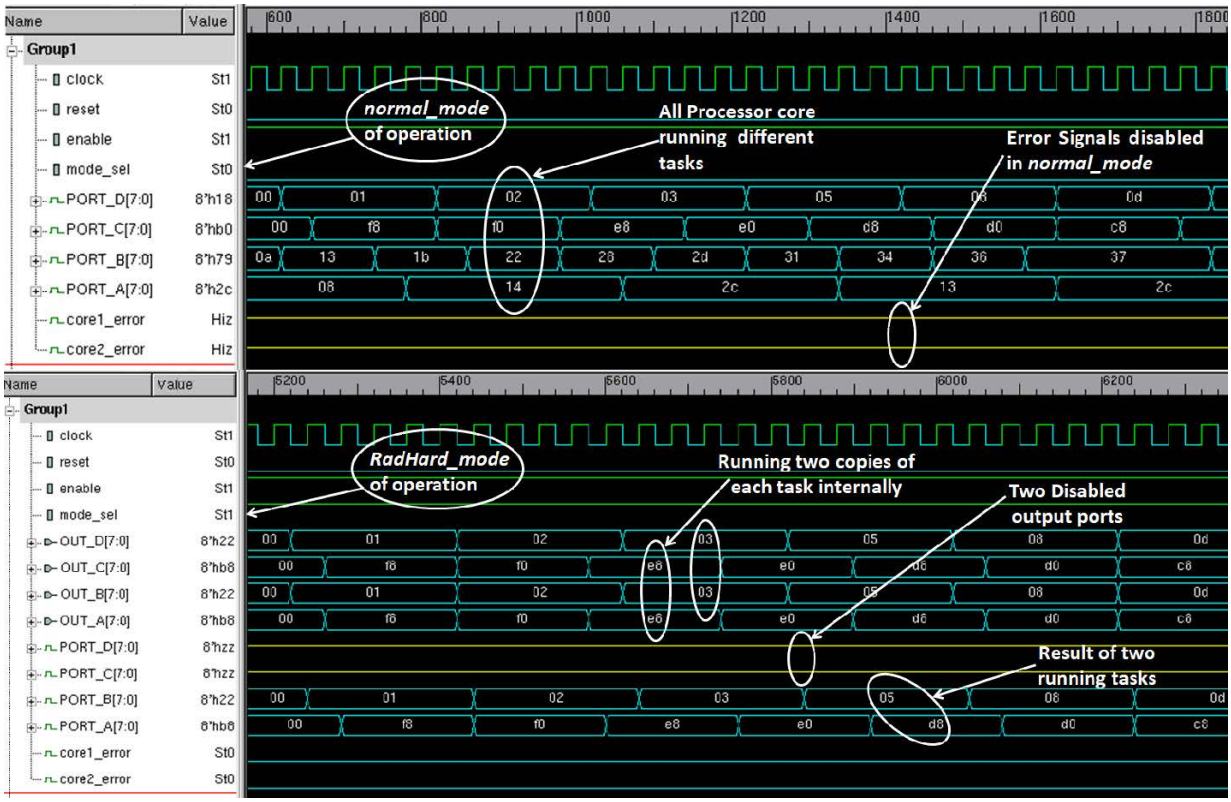

FIGURE 4. Post-synthesis simulation showing operations in normal and fault-tolerant mode for ASIC implementation of proposed architecture.

immediate values coming from the instructions. The ALU performs all the arithmetic and logical operations. As shown in figure3, all the ALU operations are performed either on two general-purpose registers or one general-purpose register and one immediate value. The store block is responsible for storing the content from the register file in data memory. Similarly, the load block loads the content of a memory location to the register file.

IV. SIMULATION, IMPLEMENTATION AND EVALUATION

The proposed fault-tolerant quad-core processor presented in figure1 is implemented using Verilog HDL. The design is synthesized using 32nm CMOS process technology node. Post-synthesis simulation, FPGA prototype, and protection evaluation using SEM (soft error mitigation) IP core are presented in the next sub-sections.

A. ASIC IMPLEMENTATION RESOURCE EVALUATION

The proposed fault-tolerant quad-core processor is synthesized on 32nm CMOS process technology node using design_compiler EDA tool of Synopsys. The gate-level netlist obtained from the design compiler after synthesis is compiled using the VCS tool of Synopsys, and post-synthesis simulation results are presented in figure4. In the normal mode of operation, the processor executes four instructions in parallel, and hence it can provide four outputs simultaneously as shown in the upper half portion of figure4. In fault-tolerant mode, two copies of every instruction are evaluated, which is shown with the help of internal signals (OUT_A, OUT_B, OUT_C, OUT_D) in the lower half of figure4. However, the final output for instruction is generated after comparing both the copies, and hence, only two output ports are enabled in fault-tolerant mode.

The synthesis result reported by the Design Compiler tool is presented in table-2 and overheads are calculated based on the synthesis result. As presented in table-2, the area overhead for incorporating fault-tolerant feature by the proposed approach is $+17.89\%$ compared to unprotected design. For the proposed fault-tolerant approach, the synthesis tool reports -21.75% reduction in power with a time penalty of $+9.96\%$ compared to unprotected design. The design is able to achieve 41MHz frequency with a time overhead of $+9.96\%$. For the proposed design, when operated in fault-tolerant mode, the throughput will depend on the occurrence of an error (because in case of error the instruction will be re-executed). Although the throughput and power consumption of design has a strong dependence on the occurrence of error, the gain in area overhead has no dependence on it.

The resource overhead results are compared with two approaches, fault-tolerant, and improved fault-tolerant, discussed in [33]. A comparison of area, power, timing, and overhead with and without reconfigurable fault-tolerant features is presented in table-2. In both fault-tolerant architectures presented in [33], the area overhead is $+20.15\%$ which

TABLE 3. Comparison of resource overhead for FPGA prototype of proposed architecture with [15].

Resource	LEON3 Soft Processor [15]						Proposed Fault-tolerant Processor		
	Single LEON3			Two LEON3			Quad-core		
	Baseline	TMR	Overhead	Baseline	TMR	Overhead	Baseline	Fault-tolerant	Overhead
LUT	4088	16041	+74.51%	11471	35311	+67.52%	12866	19853	+35.19%
Flip-flop	1950	5850	+66.66%	6626	14426	+54.07%	8367	8425	+0.69%
Slice	1397	5401	+74.14%	4546	12746	+64.33%	1615	4341	+62.79%
DSP	1	3	+66.66%	2	6	+66.66%	16	16	0%

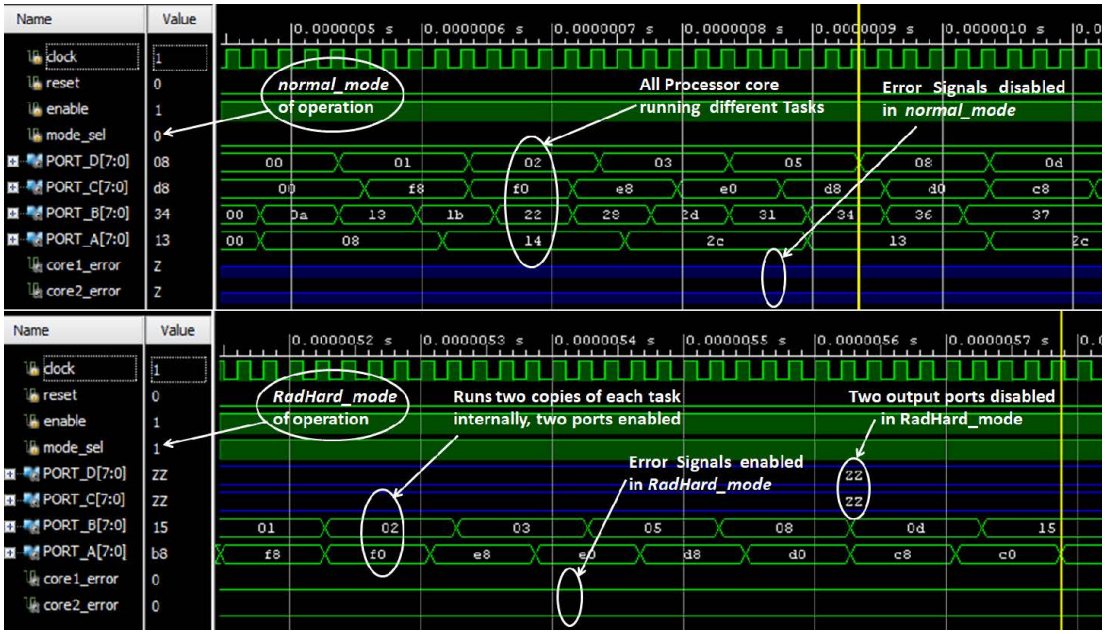

FIGURE 5. Post-implementation timing simulation of proposed architecture targeting NEXYS DDR4 FPGA board.

is reduced to +17.89% for the proposed processor. The clock period overhead for fault-tolerant and improved fault-tolerant architecture presented [33] is +54.17% and +25% respectively. For the proposed processor architecture clock period overhead is only +9.96%. Since the time overhead is more in SHAKTI-F, it achieves a low power overhead compared to the proposed design.

B. FPGA IMPLEMENTATION RESOURCE EVALUATION

To validate the functionality on silicon, the proposed fault-tolerant architecture is prototyped on a commercially available Artix-7 FPGA chip using NEXYS4 DDR FPGA board. The design of the proposed processor attains a maximum frequency of 32MHz on the Artix-7 FPGA chip. The post-implementation simulation, including the timing delay model for the FPGA prototype of the proposed design, is presented in figure5. The upper half of the figure5 shows the normal mode of operation in which the mode select signal is '0' and all four processor core runs different tasks. When the processor runs as a simple quad-core processor, the error signals core1_error and core2_error are disabled, and the processor can not detect any fault that occurs. When the mode select signal is '1', the processor works in fault-tolerant mode. Hence, the processor would run only two tasks simultaneously, as shown in the lower half of the figure5.

The resource utilization and overheads of the proposed design are compared with single-core LEON3 and two core LEON3 soft processors presented in [15]. Comparison results of the FPGA prototype are summarized in table-3. Due to its reconfigurable feature, fault tolerance for the proposed processor comes at the cost of minimal hardware overhead. The LUT (look-up table) overhead in single LEON3 processor implementation is +74.51% and it is reduced to +67.52% in two LEON3 processor implementations. The LUT overhead for the proposed processor is +35.1% which is less than both implementations presented in [15]. Since the proposed design does not use any extra core to bring fault-tolerant features (it reconfigures the existing resources) the overhead in the DSP (digital signal processor) block is Zero and it has negligible overhead in the number of flip-flops for the same reason. Although the overhead in the number of slices for the proposed design is +62.79%, it is still less than both single LEON3 processor (+74.14%), and two LEON3 processor (+64.33%) presented in [15]. Hence, for the proposed design, the overhead is less compared to the fault-tolerant approach discussed in [15].

C. ON-CHIP VALIDATION AND FAULT-TOLERANCE EVALUATION

The proposed processor is prototyped on a commercially available NEXYS DDR4 FPGA board for on-chip

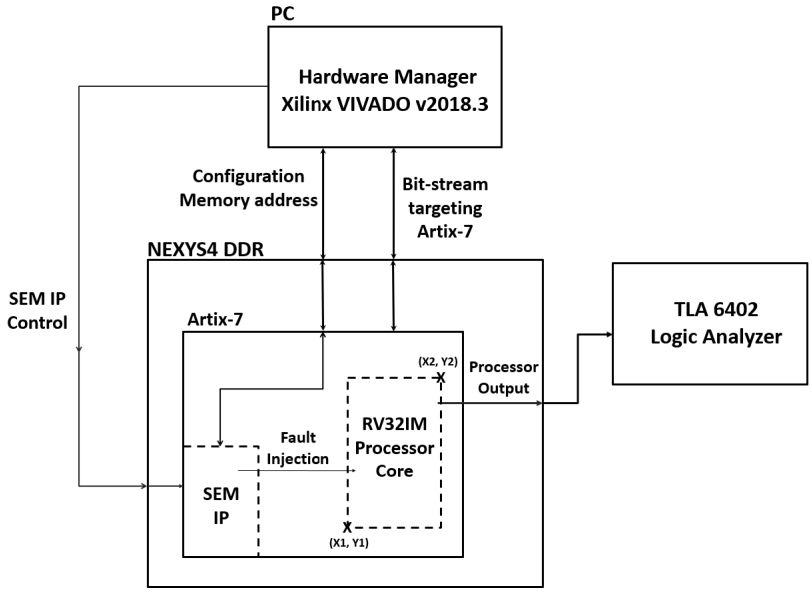

FIGURE 6. Experimental setup for fault injection using SEM IP on FPGA.

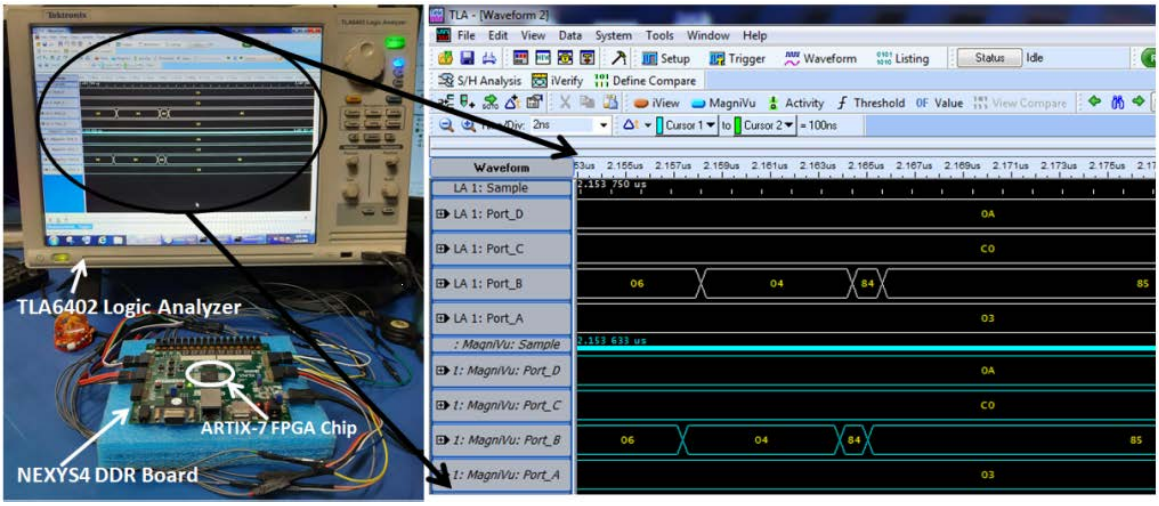

FIGURE 7. FPGA prototype for fault-injection and chip level verification in laboratory experimental setup.

validation and fault-tolerance evaluation. The proposed processor is synthesized, implemented, and loaded on the Artix-7 XC7A100T-CSG324 chip using the Vivado Integrated Design Environment (IDE) tool of Xilinx. Several algorithms are validated as part of on-chip validation. To evaluate the fault-tolerance of the proposed design a fault injection campaign has been conducted. For injecting the fault in the proposed design on FPGA, Single Event Mitigation (SEM) IP from Xilinx Inc. is used.

1) PROTECTION EVALUATION IN SIMULATION

For evaluation of fault tolerance in functional or post-synthesis simulation, an internal bit may be forced to toggle for a reasonable period of time (preferably more than the clock period to ensure injection) and it would cause a single bit-flip. Hence, fault injection may be mimicked in simulation, and the fault tolerance of the design can be tested. Moreover, since the design reevaluates the current instruction in case of error detection, there will be no change while dealing with double-bit errors. For double bit errors also two cores will provide different results and the fault detection and correction unit will generate an error signal. However, there is a possibility that in case of double-bit error two processor cores may provide exactly the same wrong result. In this case, this architecture will fail to provide fault-tolerant results.

2) SOFT ERROR MITIGATION IP CORE

Soft Error Mitigation core [39] is an IP provided by Xilinx Inc. to protect the designs against single event upset. The SEM IP core can be used in 2 modes; either for error injection

and correction or it may be used only for error injection. Here SEM core is used to inject bit-flip to test the fault-tolerance of the proposed design. This tool is selected to emulate the fault for the proposed design as it is designed to work with Xilinx products, is free, and requires only a serial interface to work. In the injection state, the SEM IP receives the frame and injects the fault into the frame which changes the configuration memory and returns again to the observation state.

3) FAULT INJECTION AND PROTECTION EVALUATION

In order to evaluate the fault tolerance of the proposed architecture, the SEM IP core of Xilinx Inc. discussed in the previous sub-section is used for fault injection. The proposed design with SEM IP core is loaded to NEXYS4 DDR board, having an XC7A100T-1CSG324C Xilinx Artix-7 FPGA. To emulate fault on the FPGA chip, an internal bit where design is mapped, needs to be flipped. The location inside the FPGA chip where a sub-blocks is to be placed may be provided using Vivado placement constraints. This allows the evaluation to be effective as one can inject error to a sensitive node which affects the functionality of the processor. The fault injection and evaluation setup are shown in figure6. The benchmark programs to evaluate fault-tolerance are selected considering certain characteristics which follow [40].

1) *Matrix multiplication:* It involves a lot of arithmetical operations that use more hardware during program execution and makes the process susceptible to error. The inputs are two 3×3 square matrices filled with random numbers.
2) *Quicksort:* The input is 30 elements unsorted array with random numbers. The input array is sorted and sorted input array is output. It involves many jumps and conditional instruction hence better for evaluating the fault tolerance.
3) *Correlation:* It is used in signal processing and involves many arithmetic and jump instructions. Hence, suitable for the evaluation of fault tolerance.
4) *Dijkstra:* This algorithm is used to find the shortest path between nodes in a graph.

The FPGA prototype in the laboratory experimental setup is shown in figure 7. For real-time evaluation and analysis, the proposed design with SEM IP core is loaded on the FPGA board. Fault Injection control and configuration memory address (fault injection address) are sent to FPGA by PC-host through Xilinx Vivado tool and the Tektronix logic analyzer(TLA6402) is connected to the NEXYS DDR4 FPGA board to verify the output result. In total, 5000 errors covering the entire range of the proposed processor are injected per benchmark. The configuration memory addresses for fault injection are generated using the ACME tool discussed in [41]. The ACME tool takes the EBD file and coordinates [(X1, Y1), (X2, Y2)] of design on FPGA as inputs and it provides the injection addresses to the SEM IP core. For the performed fault injection campaign in fault-tolerant mode, no error is found in the output result. Hence, In fault-tolerant mode, the proposed architecture provides 100 percent fault tolerance for the performed fault-injection campaign. Further, the same injection campaign is applied in the normal mode of operation for which 12 percent of errors are detected in output results.

V. CONCLUSION

In this paper, a new low-overhead DMR based reconfigurable quad-core processor architecture is proposed along with instruction scheduling algorithm for fault-tolerant applications. RISC-V ISA-based RV32IM processor core is designed and used to implement the reconfigurable quad-core processor. The reconfigurable feature makes the proposed architecture energy-efficient by optimally using all four processor cores to provide fault-tolerant results. The designed processor architecture is implemented in Verilog HDL and synthesized targeting 32nm CMOS process technology node. For fault-tolerant design, 32nm CMOS implementation result shows a significant reduction in power with a slight increase in area and time compared to unprotected design. Hence, the proposed new RISC-V-based processor architecture shall be suitable for low-overhead fault-tolerant computing systems. Further, the proposed architecture is prototyped and validated on a commercially available Artix7 FPGA chip with fault injection in laboratory experimental setup. For the proposed architecture, the resource overhead of the FPGA prototype is less compared to state-of-the-art fault-tolerant systems. The proposed design provides 100 percent fault tolerance for fault injection campaigns on FPGA. Future work will consider delaying the execution of redundant copies in the proposed design. It will create an opportunity to cancel the execution of redundant copies in case of early or successful completion of the task which will reduce the power consumption.

REFERENCES

[1] S. Safari, M. Ansari, H. Khdr, P. Gohari-Nazari, S. Yari-Karin, A. Yeganeh-Khaksar, S. Hessabi, A. Ejlali, and J. Henkel, "A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues," *IEEE Access*, vol. 10, pp. 12229–12251, 2022.
[2] S. Safari, M. Ansari, M. Salehi, and A. Ejlali, "Energy-budget-aware reliability management in multi-core embedded systems with hybrid energy source," *CSI J. Comput. Sci. Eng.*, vol. 15, no. 2, pp. 31–43, 2018.
[3] N. R. Ch, B. Gupta, and G. Kaushal, "Single-event multiple effect tolerant RHBD14T SRAM cell design for space applications," *IEEE Trans. Device Mater. Rel.*, vol. 21, no. 1, pp. 48–56, Mar. 2021.
[4] R. González-Toral, P. Reviriego, J. A. Maestro, and Z. Gao, "A scheme to design concurrent error detection techniques for the fast Fourier transform implemented in SRAM-based FPGAs," *IEEE Trans. Comput.*, vol. 67, no. 7, pp. 1039–1045, Jul. 2018.
[5] S. Pal, D. D. Sri, W. H. Ki, and A. Islam, "Highly stable low power radiation hardened memory-by-design SRAM for space applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 6, pp. 2147–2151, Jun. 2021.
[6] G. C. Messenger and M. S. Ash, "Single event phenomena I," in *Single Event Phenomena*. Boston, MA, USA: Springer, 1997, pp. 179–231.
[7] C. Claeys and E. Simoen, *Radiation Effects in Advanced Semiconductor Materials and Devices*, vol. 57. New York, NY, USA: Springer, 2013.
[8] H. Koons, J. Mazur, R. Selesnick, J. Blake, and J. Fennell, "The impact of the space environment on space systems," Aerospace Corp. El Segundo CA El Segundo Tech. Oper., Los Angeles, CA, USA, Tech. Rep. TR-99(1670)-1, 1999.

[9] R. Devaraj and A. Sarkar, "Resource-optimal fault-tolerant scheduler design for task graphs using supervisory control," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7325–7337, Nov. 2021.
[10] C. M. Krishna, "Fault-tolerant scheduling in homogeneous real-time systems," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–34, Apr. 2014, doi: 10.1145/2534028.
[11] R. Devaraj, A. Sarkar, and S. Biswas, "Fault-tolerant preemptive aperiodic RT scheduling by supervisory control of TDES on multiprocessors," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 3, pp. 1–25, Jul. 2017, doi: 10.1145/3012278.
[12] S. Mashimo, K. Inoue, R. Shioya, A. Fujita, R. Matsuo, S. Akaki, A. Fukuda, T. Koizumi, J. Kadomoto, H. Irie, and M. Goshima, "An open source FPGA-optimized out-of-order RISC-V soft processor," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2019, pp. 63–71.
[13] L. A. C. Benites, F. Benevenuti, A. B. De Oliveira, F. L. Kastensmidt, N. Added, V. A. P. Aguiar, N. H. Medina, and M. A. Guazzelli, "Reliability calculation with respect to functional failures induced by radiation in TMR arm cortex-M0 soft-core embedded into SRAM-based FPGA," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 7, pp. 1433–1440, Jul. 2019.
[14] A. Ramos, A. Ullah, P. Reviriego, and J. A. Maestro, "Efficient protection of the register file in soft-processors implemented on Xilinx FPGAs," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 299–304, Feb. 2018.
[15] A. M. Keller and M. J. Wirthlin, "Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 64, no. 64, pp. 519–528, Jan. 2017.
[16] V. Petrović, G. Schoof, and Z. Stamenković, "Fault-tolerant TMR and DMR circuits with latchup protection switches," *Microelectron. Rel.*, vol. 54, no. 8, pp. 1613–1626, Aug. 2014.
[17] S. Shukla and K. C. Ray, "Design and ASIC implementation of a reconfigurable fault-tolerant ALU for space applications," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES) (Formerly iNiS)*, Dec. 2019, pp. 156–159.
[18] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1435–1441, Aug. 2017.
[19] M. Augustin, M. Gössel, and R. Kraemer, "Implementation of selective fault tolerance with conventional synthesis tools," in *Proc. 14th IEEE Int. Symp. Design Diag. Electron. Circuits Syst.*, Apr. 2011, pp. 213–218.
[20] M. Ansari, A. Yeganeh-Khaksar, S. Safari, and A. Ejlali, "Peak-power-aware energy management for periodic real-time applications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 779–788, Apr. 2020.
[21] S. Safari, S. Hessabi, and G. Ershadi, "LESS-MICS: A low energy standby-sparing scheme for mixed-criticality systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4601–4610, Dec. 2020.
[22] H.-M. Pham, S. Pillement, and S. J. Piestrak, "Low-overhead fault-tolerance technique for a dynamically reconfigurable softcore processor," *IEEE Trans. Comput.*, vol. 62, no. 6, pp. 1179–1192, Jun. 2013.
[23] M. Ansari, S. Safari, A. Yeganeh-Khaksar, M. Salehi, and A. Ejlali, "Peak power management to meet thermal design power in fault-tolerant embedded systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 161–173, Jan. 2019.
[24] M. Salehi, A. Ejlali, and B. M. Al-Hashimi, "Two-phase low-energy N-Modular redundancy for hard real-time multi-core systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1497–1510, May 2016.
[25] S. Safari, H. Khdr, P. Gohari-Nazari, M. Ansari, S. Hessabi, and J. Henkel, "TherMa-MiCs: Thermal-aware scheduling for fault-tolerant mixed-criticality systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1678–1694, Jul. 2022.
[26] S. Budi, P. Gupta, K. Varghese, and A. Bharadwaj, "A RISC-V ISA compatible processor IP for SoC," in *Proc. Int. Symp. Devices, Circuits Syst. (ISDCS)*, Mar. 2018, pp. 1–5.
[27] J. Hennessy, N. Jouppi, S. Przybylski, C. Rowen, T. Gross, F. Baskett, and J. Gill, "MIPS: A microprocessor architecture," *ACM SIGMICRO Newslett.*, vol. 13, no. 4, pp. 17–22, 1982.
[28] D. L. Weaver, *The SPARC Architecture Manual*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
[29] C. Domas, "Breaking the x86 ISA," Black Hat, Los Vegas, NV, USA, Tech. Rep., 2017. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Domas-Breaking-The-x86-Instruction-Set-wp.pdf
[30] A. Waterman, Y. Lee, R. Avizienis, D. A. Patterson, and K. Asanovic, "The RISC-V instruction set manual volume II: Privileged architecture version 1.9," Dept. EECS, Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-129, 2016.
[31] D. Patterson and A. Waterman, *The RISC-V Reader: An Open Architecture Atlas*. Berkeley, CA, USA: Strawberry Canyon, 2017.
[32] N. Gala, A. Menon, R. Bodduna, G. S. Madhusudan, and V. Kamakoti, "SHAKTI processors: An open-source hardware initiative," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, Jan. 2016, pp. 7–8.
[33] S. Gupta, N. Gala, G. S. Madhusudan, and V. Kamakoti, "SHAKTI-F: A fault tolerant microprocessor architecture," in *Proc. IEEE 24th Asian Test Symp. (ATS)*, Nov. 2015, pp. 163–168.
[34] A. Ramos, R. G. Toral, P. Reviriego, and J. A. Maestro, "An ALU protection methodology for soft processors on SRAM-based FPGAs," *IEEE Trans. Comput.*, vol. 68, no. 9, pp. 1404–1410, Mar. 2019.
[35] F. Mireshghallah, M. Bakhshalipour, M. Sadrosadati, and H. Sarbazi-Azad, "Energy-efficient permanent fault tolerance in hard real-time systems," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1539–1545, Oct. 2019.
[36] Z. Gao, L. Zhang, T. Yan, K. Guo, Z. Xu, and P. Reviriego, "Design of SEU-tolerant turbo decoders implemented on SRAM-FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 12, pp. 2563–2572, Dec. 2020.
[37] A. Ramos, J. A. Maestro, and P. Reviriego, "Characterizing a RISC-V SRAM-based FPGA implementation against single event upsets using fault injection," *Microelectron. Rel.*, vol. 78, pp. 205–211, Nov. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026271417304493
[38] K. Chapman, "PicoBlaze 8-bit microcontroller for virtex-E and spartan-II/IIE devices," Xilinx, San Jose, CA, USA, Appl. Notes, 2003.
[39] *PG036-Soft Error Mitigation Controller V4. 1, Product Guide*, Xilinx, San Jose, CA, USA, 2015.
[40] H. Quinn, W. H. Robinson, P. Rech, M. Aguirre, A. Barnard, M. Desogus, L. Entrena, M. Garcia-Valderas, S. M. Guertin, D. Kaeli, F. L. Kastensmidt, B. T. Kiddie, A. Sanchez-Clemente, M. S. Reorda, L. Sterpone, and M. Wirthlin, "Using benchmarks for radiation testing of microprocessors and FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 62, no. 6, pp. 2547–2554, Dec. 2015.
[41] L. A. Aranda, A. Sánchez-Macián, and J. A. Maestro, "ACME: A tool to improve configuration memory fault injection in SRAM-based FPGAs," *IEEE Access*, vol. 7, pp. 128153–128161, 2019.

SATYAM SHUKLA (Student Member, IEEE) received the B.Tech. and M.Tech. degrees, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Indian Institute of Technology Patna, India. Before joining his current position, he worked as an Assistant Professor with the Department of Electronics and communication Engineering, Buddha Institute of Technology Gorakhpur, India. His research interests include processor design, very large-scale integration (VLSI) architectural design, digital VLSI design, FPGA-based system design, ASIC implementations, and system-on-chip design.

KAILASH CHANDRA RAY (Member, IEEE) received the B.E. degree in electrical engineering from the Orissa Engineering College, Bhubaneswar, India, in 1997, the M.Tech. degree in electrical engineering from NIT, Jamshedpur, India, in 2000, and the Ph.D. degree in electronics and electrical communication engineering from IIT Kharagpur, India, in 2009. He was a Technical Member Staff (Design Engineer) with CMOS Chips, Bengaluru, India, from 2001 to 2003. He was also a Lecturer with IIIT Allahabad, India, from 2008 to 2010. He was an Assistant Professor with the Department of Electrical Engineering, Indian Institute Technology Patna, India, from 2010 to 2018, where he has been an Associate Professor, since 2018. His research interests include very large-scale integration (VLSI) architectural design, VLSI signal processing, digital VLSI design, hardware design methodologies, FPGA-based system design, CORDIC, and embedded systems.