



Low Power and Radiation Tolerance System on Chip (Soc) Architecture

Misbah Husain¹, Prof Shivraj Singh²

Research Scholar, Department of Electronics and Communication, Sagar Institute of
Research and Technology, Bhopal¹

Assistant Professor, Department of Electronics and Communication, Sagar Institute of
Research and Technology, Bhopal²

Abstract

The design and assessment of a low-power, radiation-tolerant System-on-Chip (SoC) architecture appropriate for RTL-level VLSI implementation was the main goal of the current work. Instruction memory, ECC-protected data memory, a memory-mapped interface, UART, GPIO, timer, interrupt controller, watchdog timer, Power Management Unit (PMU), and Fault-Tolerance Controller (FTC) were all incorporated into a small 32-bit RV32I-compatible processing subsystem. Practical methods for FPGA-oriented RTL design featured peripheral sleep modes, operand isolation, synchronous clock-enable control, memory-enable management, and a decrease in pointless bus activity. Instead of using full-chip redundancy, radiation tolerance was added selectively. The provided dataset showed a 14.6% drop in dynamic power from 82.4 mW to 70.4 mW and a 9.1% decrease in overall power from 101.0 mW to 91.8 mW. LUT use rose from 4,920 to 6,835 and flip-flop usage from 3,280 to 4,725. The claimed maximum operational frequency dropped from 109.9 MHz for the baseline to 103.6 MHz, while the combined design remained above the selected 100 MHz threshold. There were 1,800 injected events in the specified RTL fault-injection campaign; 1,786 events were detected and 1,207 events were categorised as effectively mitigated, representing 99.2% detection and 67.1% successful mitigation.

Keywords: System-on-Chip (SoC), RISC-V RV32I, low-power design, selective TMR, SECDED ECC, fault injection, FPGA, radiation tolerance

1. INTRODUCTION

The fast growth of Very Large-Scale Integration (VLSI) technology has altered the design and implementation of digital systems. Functions that formerly needed several integrated circuits, printed-circuit board interconnections and separate control devices may now be integrated on a single silicon substrate. This degree of integration has led to the widespread adoption of the System-on-Chip (SoC) where processor, memory, communication, timing, control, and peripheral functions are integrated into a single coordinated architecture. Now SoCs are utilised in portable electronics, automobile control units, industrial systems, medical instruments, communication equipment, aerospace electronics and a lot of other embedded applications where small size, high performance, low energy consumption and reliable operation are crucial.

1.1 Background of System-on-Chip Technology

A System-on-Chip (SoC) is an integrated digital system where the main functional components necessary for a given application are integrated on a single chip or programmable fabric. SoCs may have one or more processor cores, instruction and data memories, buses, direct-memory-access units, timers, interrupt logic, serial interfaces, analogue or mixed-signal blocks, accelerators, security logic, and power-management circuits, depending on the application. The main characteristic of a SoC is not so much the number of blocks on chip but the integrated coordination of compute, storage, communication and control under a shared architecture.

1.1.1 RISC-V as a Processing Core for the Proposed SoC

The processor used for the proposed architecture is a 32-bit RISC-V RV32I core. The RISC-V instruction-set architecture is appropriate for academic and research focused VLSI implementation as its base integer instruction set is small, open and modular. The RV32I foundation provides the arithmetic, logical, branch, jump, load, and store operations necessary for a tiny, embedded processor, without the need for massive optional instruction-set expansions. The simplicity of the processor hardware allows integration and fault-injection investigation at RTL-level.

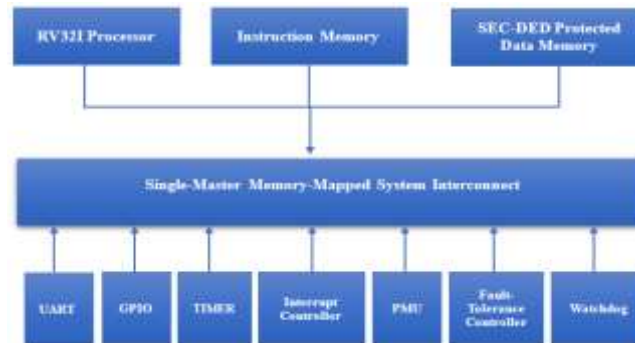


Figure 1.1: Conceptual organization of the proposed low-power and radiation-tolerant SoC architecture

1.2 Low-Power Design in VLSI Systems

Low power design is one of the major needs of current VLSI systems since energy consumption influences battery life, thermal behaviour, dependability, packaging and the total system cost. SoC may satisfy functional and temporal constraints, but if it consumes too much power, it may not be viable for portable or thermally limited applications. Therefore, the suggested design implements power-management features at RTL level, rather than only depending on manufacturing technologies to decrease power.

1.2.1 Dynamic Power Dissipation

Dynamic power is the part of power that is mostly caused by logic changes. Whenever a CMOS node is switching between logical states, the capacitances of gates, connectors and loads are charged or discharged. The first order representation of widely used switching power is:

$$P_{\text{dyn}} = \alpha C_{\text{eff}} V_{\text{DD}}^2 f \quad (1.1)$$

where α is the switching activity factor, C_{eff} is the effective load capacitance, V_{DD} is the supply voltage and f are the operating frequency. Equation considers the tremendous issue of unneeded switching. With fixed voltage and technology, the dynamic energy may be reduced immediately by decreasing the activity factor. The assessed FPGA project uses a fixed 100 MHz clock. Frequency scaling is not implemented and requires future work.

1.3 Research Objectives

The study is carried out with the following specific objectives:

- To design a low-power System-on-Chip architecture suitable for RTL simulation and an FPGA-oriented implementation flow.
- To integrate radiation fault-detection and correction mechanisms into the SoC without using full-chip Triple Modular Redundancy.
- To reduce dynamic power using clock-enable control, activity reduction, operand isolation, memory enable control, and idle-module disabling.
- To protect critical registers, memories, counters, and control paths from simulated radiation-induced single-bit and transient faults.

- To analyse the power, area, timing, recovery latency, and fault-coverage overhead introduced by low-power and reliability logic.
- To compare the proposed protected SoC with conventional unprotected SoC under common workload, testbench, and implementation conditions.

2. RESEARCH METHODOLOGY AND SYSTEM SPECIFICATIONS

The methods of the study and the requirements of the system used to develop low-power and radiation-tolerant System-on-Chip (SoC) architecture are presented in this chapter. The proposed approach is presented as an RTL-level VLSI design technique in which a typical baseline SoC is first built, validated and synthesised. Low power approaches and radiation fault-tolerance measures are then judiciously incorporated, so that the final design may be validated against the same workload, tool flow and technology assumptions.

Consequently, in this chapter, the issue formulation, research goals, baseline SoC specification, processor selection reasoning, low-power design strategy, radiation fault model, mitigation strategies, development tools, target platform and evaluation parameters are defined. The focus is on repeatability, each architectural change is measured using quantifiable measures like power, area, timing, recovery delay, fault coverage.

2.1 Research Problem Formulation

One key challenge is dependability under radiation impacts. High energy particles may upset charge stored in flip-flops, SRAM cells, counters, status registers, and program-control registers. In the abstraction level employed in this study, RTL, such occurrences are described as bit-flips or transient disturbance of control-signals. If the upset is in a general purpose data register the mistake may damage a computation. Effects on the programme counter, interrupt controller, timer, watchdog, bus state machine, or fault-status register may spread into system-level dysfunction.

2.2 Proposed Research Methodology

The research process is based on a step-by-step design flow. First the baseline SoC is defined and constructed to assess its usual functionality, power, timing and area. The suggested SoC is then developed by adding low power control and radiation-mitigation blocks to the same functional design. This technique allows a fair comparison as the protected and baseline designs have the same CPU, bus interface, memory, peripherals, workloads and target technology assumption.

1. Requirement analysis: Define the required processor, memory map, peripherals, operating frequency, target platform, fault assumptions, and evaluation metrics.
2. Baseline SoC design: Create a conventional unprotected SoC containing the selected processor, bus fabric, instruction memory, data memory, UART, GPIO, timer, interrupt controller, watchdog, and reset unit.
3. RTL coding: Implement the baseline and proposed modules using Verilog HDL or SystemVerilog with synthesizable coding practices.
4. Functional verification: Verify processor execution, memory access, bus transactions, peripheral control, interrupt handling, timer operation, and watchdog recovery using simulation testbenches.



Figure 2.1: Overall research methodology flowchart

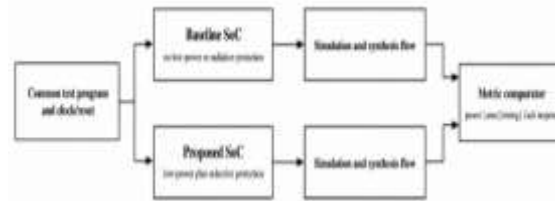


Figure 2.2: Baseline and proposed design comparison flow

2.3 Low-Power Design Methodology

The low-power design process is centred on minimising needless switching activity while maintaining the functional functionality of the SoC. Power in CMOS digital circuits is often classified as dynamic power and static power. Dynamic power is linked with the charging/discharging of the capacitance for the transitions of the signals, while static power is mostly involved with the leakage currents when the transistors are not switching. The suggested technique is mainly focused on dynamic power since it can be directly controlled at RTL level via clock and activity control.

Dynamic Power Consumption

Dynamic power is spent during charging and discharging of circuit nodes. The single effective activity-factor convention in Eq. (3.1) ensures that no extra one-half factor is added elsewhere. At RTL, clock enables, memory enables, operand isolation and bus quiescence mostly decrease α , voltage and capacitance being implementation-dependent. [1]

$$P_{\text{dyn}} = \alpha C_{\text{eff}} V_{\text{DD}}^2 f \quad (2.1)$$

where α is effective transition activity per clock cycle, C_{eff} is effective switched capacitance, V_{DD} is supply voltage and f is clock frequency. The reported FPGA-oriented study uses a fixed 100 MHz clock; dynamic voltage or frequency scaling is not part of the evaluated implementation.

2.4 Radiation-Mitigation Methodology

The radiation mitigation approach is based on selected redundancy, error detection, correction and controlled recovery. The suggested design protects modules according to their criticality, rather than triplication of the whole SoC. A problem in a non-critical data register may be noticed and documented, but a fault in the programme counter, interrupt controller, watchdog, or power-management state machine may need quick rectification or recovery. This selected approach decreases the size and power overhead compared to full-chip TMR.

2.5 Development Tools

The proposed SoC architecture was developed and evaluated using a standard RTL-to-FPGA design methodology. System Verilog was used for RTL description of the processor, memory subsystem, power-management logic, fault-tolerance mechanisms, peripherals, and supporting control modules. Functional verification and fault-injection experiments were performed using simulation-based testbenches, while FPGA synthesis, implementation, timing analysis, and power estimation were carried out using the Vivado/XSim 2026.1 design environment. The evaluation methodology maintains a common processor configuration, clock constraint, workload assumption, and architectural interface across the baseline, low-power, radiation-protected, and proposed combined SoC variants so that power, timing, resource utilization, and fault-response characteristics can be compared consistently. The implementation flow also supports generation of RTL schematics, simulation waveforms, utilization summaries, timing information, and power-analysis results for architectural evaluation. [6] [7]

2.6 Target FPGA or Technology Library

The quantitative implementation of the proposed architecture is targeted to an Artix-7-class FPGA using the Vivado/XSim 2026.1 design environment. For consistency throughout the study, the FPGA device considered for synthesis and implementation is xc7a200tfbg676-1, with a primary operating-clock constraint of 100 MHz corresponding to a 10.00 ns clock period. The same target device, timing constraint, architectural configuration, and evaluation methodology are considered for the baseline, low-power, radiation-protected, and proposed combined SoC variants. The quantitative evaluation is restricted to FPGA-oriented parameters including LUT utilisation, flip-flop utilisation, BRAM usage, timing, operating frequency, and power consumption. ASIC gate count, physical silicon area, PVT-corner analysis, and post-layout physical-design results are outside the scope of the present work.

3. PROPOSED LOW-POWER AND RADIATION-TOLERANT SOC ARCHITECTURE

The suggested low-power and radiation-tolerant System-on-Chip architecture is presented in this chapter. A 32-bit RISC-V processor core, a memory-mapped bus, instruction and data memories, common embedded peripherals, a power management unit, and a fault-tolerance controller form the foundation of the architecture. The chapter's primary contribution is not only the choice of these blocks but also the way radiation-aware protection and low-power control are combined into a single SoC fabric.

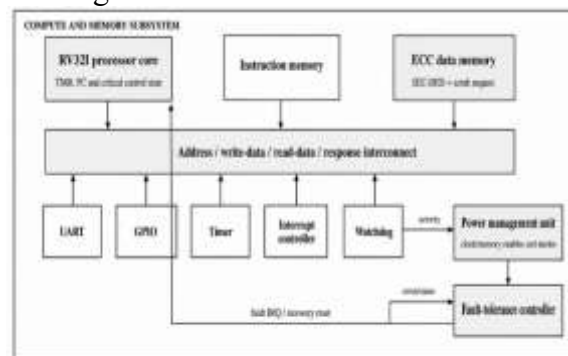


Figure 3.1: Complete proposed SoC block diagram

3.1 Top-Level SoC Architecture

The integration of radiation-tolerance and low-power features is made easier by the central interface. Bus activity may be monitored at a single location to spot problem replies, idle peripherals, recurring transactions, and underutilised memory banks. Using the same memory-mapped structure, the fault-tolerance controller may also update status registers and keep an eye on bus error replies. This prevents a dispersed control approach and enables software to use standard load instructions to obtain fault and power-mode information.

Functional and supervisory pathways are kept apart by architecture. Instruction, data, address, control, interrupt, and peripheral signals are all carried along functional routes. Clock-enable signals, module sleep requests, ECC/parity flags, TMR mismatch signals, watchdog timeout signals, recovery instructions, and safe-mode requests are all carried via supervisory routes. This division makes the suggested architecture more understandable at the RTL level and enables independent verification of each supervisory mechanism prior to full-system simulation.

3.2 Processor Core Architecture

The project-specific educational RTL block processor_core is used by the processing subsystem. It is not a brand-new or completely compliance-tested RISC-V processor; rather, it is an RV32I-compatible integration vehicle for the directed tests in Appendix F. Since the source package does not include any third-party repositories or licenses, none are claimed. The thesis contribution starts at the SoC integration border and includes the fault-injection mechanism, power-management controls, selective protection wrappers, bus and memory integration, and FTC/watchdog response. [4]

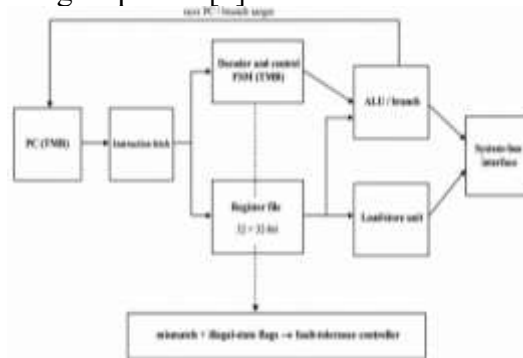


Figure 3.2: Processor internal architecture

4. RTL IMPLEMENTATION, VERIFICATION AND FAULT-INJECTION FRAMEWORK

This chapter describes the proposed low-power and radiation-tolerant SoC's register-transfer-level implementation, verification approach, and fault-injection mechanism. The RTL stage's goal is to convert the architectural choices outlined in Chapter 4 into verification structures, connection logic, protection logic, and synthesizable modules that may be simulated prior to synthesis. As a consequence, the chapter serves as a link between the quantitative synthesis, power, time, and fault-coverage assessment presented in the subsequent result chapter and the suggested design.

4.1 RTL Design Hierarchy

The top-level SoC block diagram was used to create the RTL hierarchy. The global clock, reset, external peripheral pins, interrupt outputs, and debug/status signals are all exposed by the soc_top module, which also serves as the integration boundary. Below this top module, all significant functional blocks are instantiated. The functional datapath, low-power control logic, and protection logic are all maintained apart by purposefully keeping the hierarchy short enough for readability.



Figure 4.1: RTL module hierarchy of the proposed SoC

4.2 RTL Implementation of Baseline SoC

An unprotected reference design is used to implement the basic SoC. It lacks ECC, selective TMR, parity checking, watchdog recovery control, and automated peripheral sleep management, but it makes use of the same CPU core, memory, bus address map, and peripheral modules as the suggested SoC. By keeping the functional resources shared, it is ensured that any measurable overhead in the suggested architecture is mostly driven by low-power additions and protection rather than by unrelated architectural modifications.

Every module in the baseline RTL gets the same synchronous reset and global clock. Even when no transaction is directed towards peripheral register files, they are nonetheless timed. Simple word-wide storage arrays without redundant check bits are used in memory read and write operations. Because processor state registers are single-copy registers, temporary corruption in a crucial register cannot be fixed until the programme subsequently overwrites it. This behaviour is indicative of traditional low-complexity SoC implementation.

4.3 Implementation of Clock Gating

Synchronous clock-enable logic is used in RTL to describe clock activity control. The module state only changes when the local enable is asserted; the global FPGA clock is not gated in the LUT fabric. Although it is not utilised in Chapter 6, mapping the same purpose to an ASIC integrated clock-gating cell is a potential future implementation option.

Listing 4.1: Clock-enable based RTL implementation for an idle peripheral register.

```
always_ff @(posedge clk) begin
    if (rst_sync || fault_reset_req) begin
        uart_ctrl_reg <= 32'h0000_0000;
    end else if (uart_clk_en) begin
        if (bus_wr && uart_sel && addr == UART_CTRL_ADDR) begin
            uart_ctrl_reg <= bus_wdata;
        end
    end
end
end
assign uart_clk_en = uart_sel | uart_busy | irq_pending | pmu_force_active;
```

The enable equation contains the selected bus access, internal busy flag, interrupt condition and PMU override. Therefore, the module remains active when a transaction is in progress or when an interrupt must be serviced, but it stays idle when no functional event is pending. The same design pattern is used for GPIO, timer, watchdog register access and memory enable control.



Figure 4.2 XSim behavioral simulation of synchronous clock-enable operation showing state retention during disabled cycles

4.4 Implementation of Selective TMR

Listing 4.2: Majority voter and mismatch flag used in selective TMR blocks.

```
assign voted_bit = (replica_a & replica_b) |
    (replica_b & replica_c) |
    (replica_a & replica_c);
assign mismatch = (replica_a != replica_b) |
```

```
(replica_b != replica_c) |
(replica_a != replica_c);
always_ff @(posedge clk) begin
    if (rst_sync) begin
        tmr_error_count <= '0;
    end else if (mismatch) begin
        tmr_error_count <= tmr_error_count + 1'b1;
    end
end
end
```



Figure 4.3: XSim behavioral simulation of selective-TMR fault injection showing single-replica corruption, majority-voted output retention and mismatch detection

4.5 Implementation of ECC Memory

Listing 4.3: ECC read-path concept for syndrome, correction and error flags.

```
always_comb begin
    corrected_data = read_data;
    single_error = 1'b0;
    double_error = 1'b0;
    unique case ({overall_parity_error, |syndrome|})
        2'b00: ; // no error
        2'b11: begin
            single_error = 1'b1;
            corrected_data = correct_mapped_codeword_bit(read_data, syndrome);
        end
        2'b10: single_error = 1'b1; // overall-parity-bit-only error
        2'b01: double_error = 1'b1; // detected, not corrected
    endcase
end
```

Table 4.1: ECC memory implementation signals

Signal / field	Width	Role	Verification check
Write data	32 bits	Input to ECC encoder	Seven check bits generated before write
Hamming parity	6 bits	Locates one error in the 38-bit Hamming word	Syndrome maps to a legal codeword position
Overall parity	1 bit	Distinguishes odd- and even-weight patterns	Overall-parity-only error handled explicitly
Stored codeword	39 bits	32 data + 7 check bits	Fault-free readback matches encoded write
Syndrome	6 bits	Candidate Hamming-word error position	Mapped through codeword layout; not used as raw data index

single_error	1 bit	Correctable one-bit event	Includes overall-parity-bit-only event
double_error	1 bit	Detected, uncorrectable two-bit event	No data correction attempted



Figure 4.4: XSim behavioral simulation of SEC-DED single-bit fault injection showing syndrome generation, error classification and successful data correction



Figure 4.5: XSim behavioral simulation of SEC-DED double-bit fault injection showing syndrome generation and uncorrectable-error detection

4.5 Fault-Injection Framework

For register, memory, and one-cycle control failures, the fault-injection framework specifies target, bit, cycle, duration, observation window, and result variables. An openly exposed DUT observation point is perturbed by a testbench instrumentation mask before being cleared; a selector-only task is not seen by the framework as proof that a hierarchical force/release took place. The campaign description is provided in Appendix C, and an example testbench and assertion code are provided in Appendix E. [5]

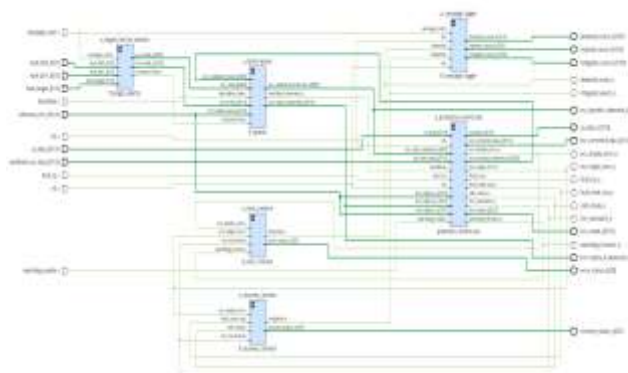


Figure 4.6: Vivado elaborated RTL schematic of the fault-injection harness showing target selection, fault injection, protection-control RTL, error/recovery monitoring and campaign logging

Listing 4.4: Example testbench task for single-bit register fault injection.

```
logic [31:0] fi_pc_mask_b;
assign pc_replica_b_observed = pc_replica_b ^ fi_pc_mask_b;

task automatic inject_pc_replica_b(input int unsigned bit_index,
                                input int unsigned duration_cycles = 1);
    assert (bit_index < 32) else $fatal("invalid injection bit");
    fi_pc_mask_b = '0;
```

```
@(negedge clk); fi_pc_mask_b[bit_index] = 1'b1;
repeat (duration_cycles) @(posedge clk);
@(negedge clk); fi_pc_mask_b = '0;
endtask
```

4.6 FPGA/ASIC Implementation Flow

Synthesizable SystemVerilog, XSim functional simulation, Vivado synthesis and implementation, static timing analysis, and activity-based power estimates are all part of the anticipated FPGA-specific implementation sequence. All four variations need the same top-level setup, target constraint, and workload. Physical design and ASIC synthesis are not included in the quantitative scope.

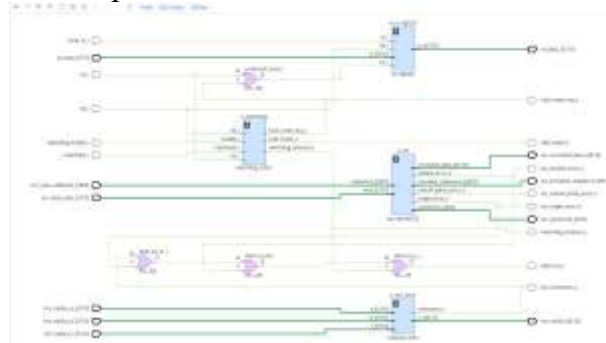


Figure 4.7: Vivado elaborated RTL schematic of the protection-control subsystem integrating clock-enable logic, selective TMR, SEC-DED ECC and watchdog recovery

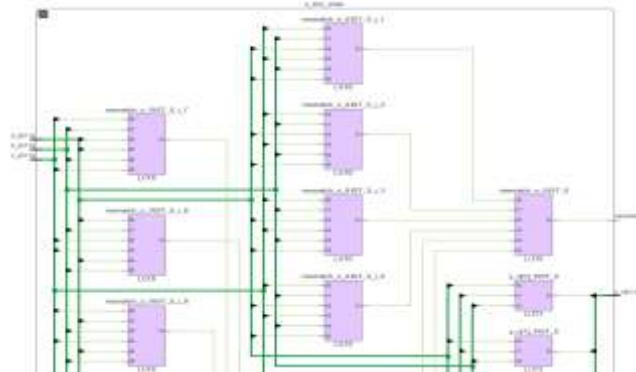


Figure 4.8: Vivado synthesized schematic of the selective-TMR majority-voter and mismatch-detection logic

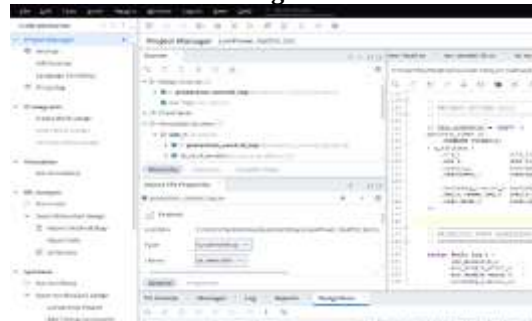


Figure 4.9: Vivado project view containing an archived protection-control subsystem run; the current XPR top differs and requires synthesis/implementation rerun

5. RESULTS, COMPARATIVE ANALYSIS AND DISCUSSION

The simulation, synthesis, timing, power, and fault-injection findings for the suggested low-power and radiation-tolerant SoC design are presented in this chapter. A standard baseline SoC,

a low-power-only SoC, a radiation-protected SoC, and the suggested combination SoC are the four design variations for which the findings are presented. The separate impact of low-power logic, the independent cost of protection logic, and the ultimate trade-off achieved when both goals are combined in a single design can all be seen thanks to this division.

5.1 Functional Simulation Results

To verify the SoC's baseline functioning, functional simulation was done prior to fault injection. Instruction memory was loaded, bus transactions were applied, memory read and write operations were practiced, the UART was setup, GPIO ports were toggled, timer events were created, and interrupt-controller behaviour was examined by the simulation testbench. Only when the observed output matched the predicted signature and no unexpected error flag was raised was a pass condition proclaimed.

Table 5.1 Functional simulation result summary

Test ID	Function	Simulation stimulus	Observed result	Status
FS-01	Processor operation	64 directed RV32I instruction programs	All final register signatures matched	Pass
FS-02	Memory read/write	2048 aligned word transactions	No data mismatch; byte-enable behavior correct	Pass
FS-03	UART operation	256 loopback bytes at configured baud rate	Transmit and receive data matched	Pass
FS-04	GPIO operation	128 output/input pattern checks	Pin state and status register matched	Pass
FS-05	Timer and interrupt	50 compare-match and overflow events	Interrupt request and clear sequence correct	Pass

Module Wake-Up Results

Table 5.2 Module wake-up latency results

Module	Wake-up source	Latency	Functional result
UART	Bus write to control/status register	2 cycles	No byte lost after wake-up
GPIO	External input change or bus access	1 cycle	Input sample updated correctly
Timer	Timer enable or compare event	1 cycle	Counter resumes from retained value
Data memory	Load/store request	1 cycle	Read data valid after enable
Interrupt controller	Interrupt request	2 cycles	Pending bit captured and prioritized

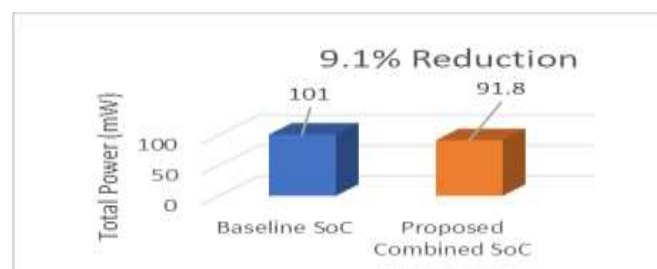


Figure 5.1: Baseline versus proposed total-power bar graph

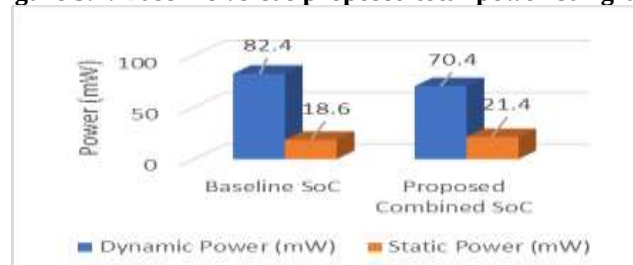


Figure 5.2: Dynamic and static power comparison

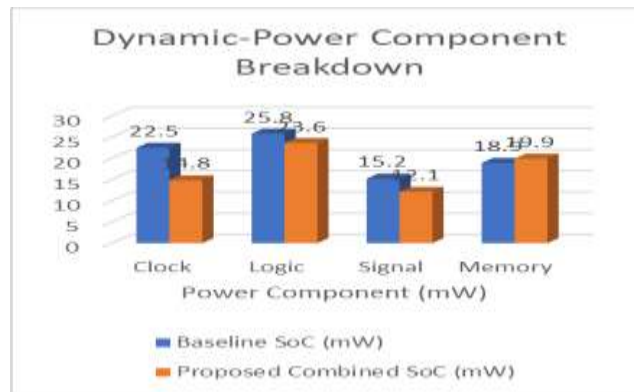


Figure 5.3: Power-component comparison between baseline and proposed combined SoC

5.2 Radiation Fault-Injection Results

At the RTL level, radiation fault-injection tests were carried out by corrupting specific registers, memory bits, and transient control signals. Unprotected register faults, TMR-protected register faults, single-bit and double-bit memory errors, watchdog recovery scenarios, and unrecoverable error answers are all included in the injection campaign. Every experiment documents the fault's detection, correction, masking, reset recovery, or reporting as an unrecoverable event.

Table 5.3: Radiation fault-injection result summary

Fault class	Injected	Detected	Successfully mitigated	Detection rate (%)	Mitigation rate (%)
Critical register with TMR	400	400	400	100.0	100.0
Non-critical parity register	250	250	0	100.0	0.0
Single-bit data memory	500	500	500	100.0	100.0
Double-bit data memory	250	250	0	100.0	0.0
Transient control signal	300	286	211	95.3	70.3
Watchdog hang/reset	100	100	96	100.0	96.0
Overall (campaign-weighted)	1,800	1,786	1,207	99.2	67.1

Unprotected Register Fault

When a program counter or control-state register was corrupted in the unprotected baseline, the fault often propagated into an invalid instruction fetch, incorrect branch target or processor hang. Some faults were indirectly detected by invalid bus activity or watchdog timeout, but the baseline design had no immediate register-level correction mechanism.



Figure 5.4: XSim behavioral simulation of an unprotected program-counter fault showing persistent state divergence and absence of automatic correction

TMR-Protected Register Fault

For TMR-protected registers, one replica was corrupted at a time while the other two replicas retained the correct value. The majority voter preserved the correct architectural output and asserted a mismatch flag. The mismatch flag was counted by the fault-tolerance controller for diagnostic reporting.



Figure 5.5: XSim behavioral simulation of selective-TMR fault masking showing single-replica corruption, majority-voted output retention, mismatch detection and fault indication

Unrecoverable Error Response

Unrecoverable errors include double-bit memory corruption and selected persistent control faults. These events were not silently ignored. Instead, the proposed SoC asserted the error interrupt, stored the error class and address where available, and entered safe mode when the fault could not be corrected locally.

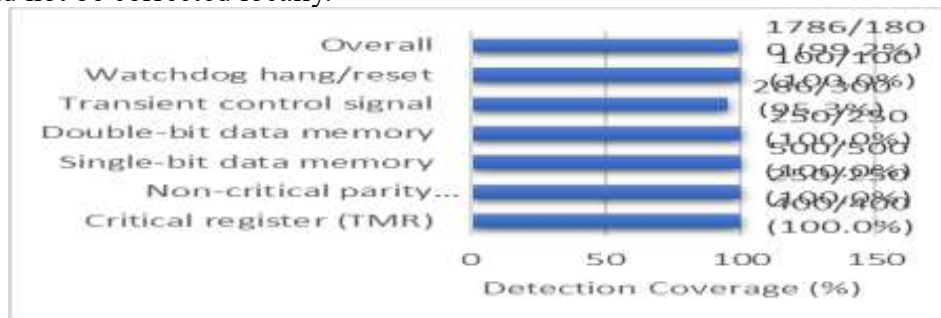


Figure 5.6: Fault-detection coverage for injected fault classes

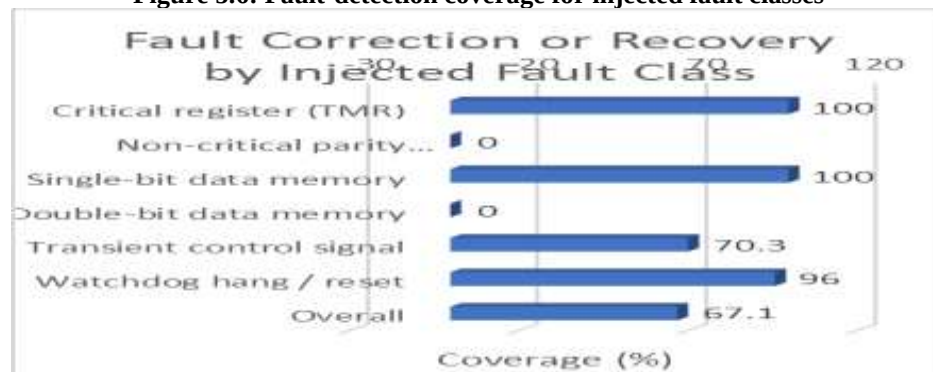


Figure 5.7: Successful-mitigation rate for the defined fault classes

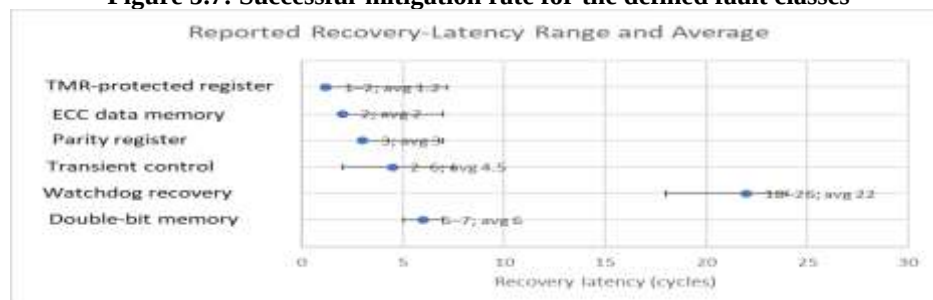


Figure 5.8: Class-specific fault-response latency

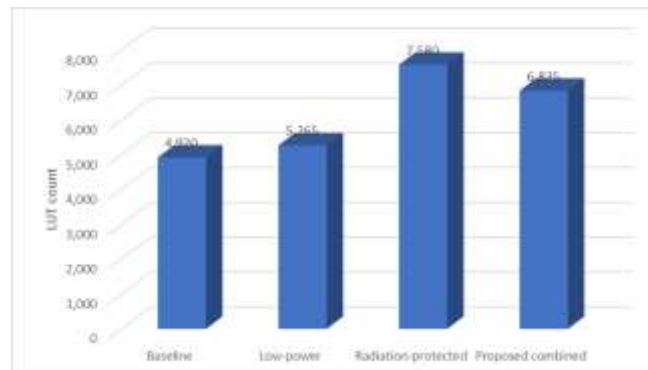
5.3 Synthesis Results

The hardware cost of adding radiation-tolerance circuitry and low-power control is quantified by the synthesis findings. The CPU core, memory interfaces, bus connectivity, UART, GPIO, timer, and interrupt controller are all included in the basic SoC. Clock-enable control, operand isolation, and memory-enable management are added by the low-power SoC. More comprehensive protection logic is added by the radiation-protected SoC. Selective TMR, ECC memory, parity-protected non-critical registers, watchdog recovery, and a shared fault-tolerance controller are all used in the suggested combined SoC.

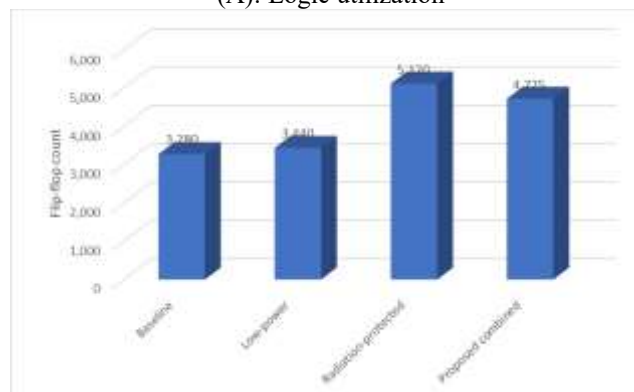
Table 5.4: Synthesis utilization comparison

FPGA resource	Baseline	Low-power	Radiation-protected	Combined
LUTs	4,920	5,265	7,580	6,835
Flip-flops	3,280	3,440	5,120	4,725
BRAMs	8	8	10	10
DSPs	1	1	1	1

The combined variant uses 6,835 LUTs and 4,725 flip-flops. Its LUT count is 38.9% above the baseline and 9.8% below the radiation-protected variant, showing the reported FPGA resource effect of selective rather than broader protection. These device-oriented counts are not converted into ASIC gates or silicon area.

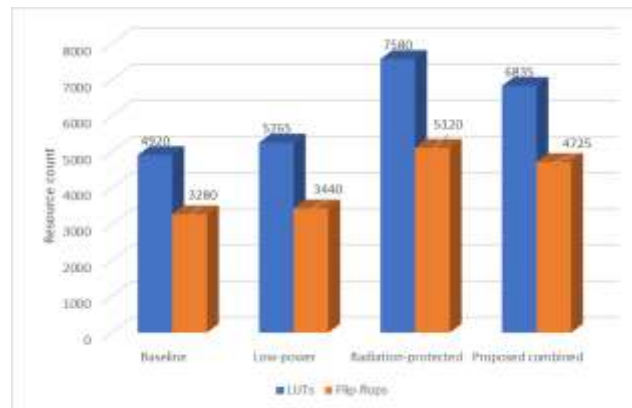


(A): Logic utilization

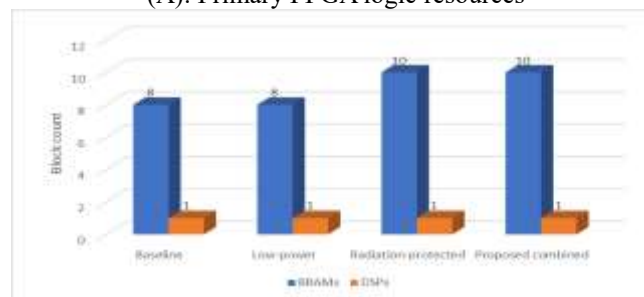


(B): Sequential-logic utilisation

Figure 5.9: FPGA LUT and flip-flop count comparison derived



(A): Primary FPGA logic resources



(B): Dedicated resources

Figure 5.10: FPGA resource-utilization comparison derived from Table 5.4

5.4 Timing Analysis

Timing uses two distinct quantities. The minimum feasible period $T_{\min}$ derives $f_{\max} = 1000/T_{\min}$ and setup slack at the 10.00 ns target as $10.00 - T_{\min}$. The separate critical data-path delay is retained only for the PDP calculation. This arithmetic normalization is not a substitute for an archived static-timing report.

Table 5.5: Timing analysis comparison.

Timing metric	Baseline	Low-power	Radiation-protected	Combined
Critical data-path delay for PDP (ns)	8.72	8.95	10.84	9.48
Minimum feasible period, $T_{\min}$ (ns)	9.10	9.35	11.22	9.65
Maximum frequency, $1000/T_{\min}$ (MHz)	109.9	106.9	89.1	103.6
Setup slack at 100 MHz, $10 - T_{\min}$ (ns)	+0.90	+0.65	-1.22	+0.35
Hold slack (ns)	+0.21	+0.19	+0.13	+0.17

The baseline minimum period is 9.10 ns (109.9 MHz) and the combined variant is 9.65 ns (103.6 MHz). At a 10.00 ns target, their setup slacks are +0.90 ns and +0.35 ns, respectively; the combined variant also has +0.17 ns hold slack. The radiation-protected variant has -1.22 ns setup slack and therefore does not meet 100 MHz in the tabulated model.

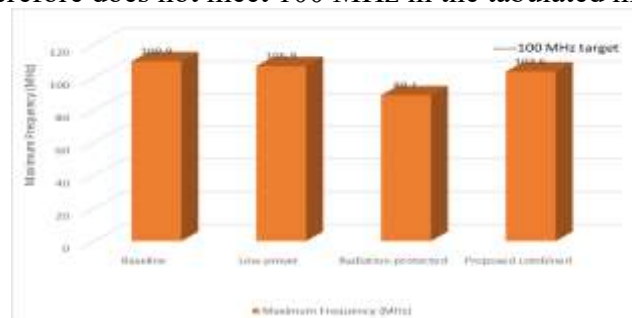
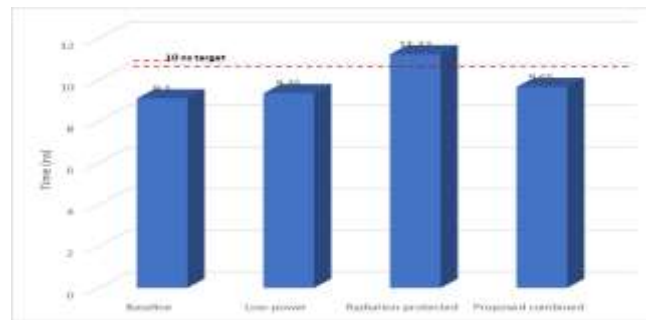
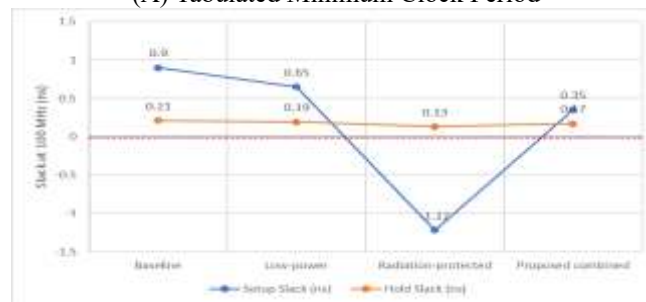


Figure 5.11: Maximum-frequency comparison derived from the minimum clock periods



(A) Tabulated Minimum Clock Period



(B) Normalized Setup and Hold Margins

Figure 5.12: Normalized timing summary derived

5.5 Power Analysis

Power analysis separates the total power into clock power, logic power, signal power, memory power, static power and dynamic power. Dynamic power includes the switching components of clock, logic, signal and memory resources. Static power represents leakage and device bias components that remain even when switching activity is low.

Table 5.6: Power analysis comparison.

Power component	Baseline SoC	Low-power SoC	Radiation-protected SoC	Proposed combined SoC
Clock power (mW)	22.5	15.1	31.4	14.8
Logic power (mW)	25.8	20.2	37.6	23.6
Signal power (mW)	15.2	10.7	21.8	12.1
Memory power (mW)	18.9	15.3	21.8	19.9
Dynamic power (mW)	82.4	61.3	112.6	70.4
Static power (mW)	18.6	18.9	22.3	21.4
Total power (mW)	101.0	80.2	134.9	91.8

Baseline Power

The baseline SoC consumed 101.0 mW total power, of which 82.4 mW was dynamic and 18.6 mW was static. Clock, logic and signal switching were the major dynamic contributors because all modules remained clocked during most of the simulation.

Proposed SoC Power

The proposed combined SoC consumed 91.8 mW total power. Although selective TMR, ECC and status logic increase the amount of hardware, the low-power controller reduces clock and signal switching enough to produce a net total-power reduction compared with the baseline.

6. CONCLUSION

The design and assessment of a low-power, radiation-tolerant System-on-Chip (SoC) architecture appropriate for RTL-level VLSI implementation was the main goal of the current



work. Reducing needless power usage and enhancing dependability against radiation-induced defects are two crucial needs of contemporary embedded systems. Instruction memory, ECC-protected data memory, a memory-mapped interface, UART, GPIO, timer, interrupt controller, watchdog timer, Power Management Unit (PMU), and Fault-Tolerance Controller (FTC) were all incorporated into a small 32-bit RV32I-compatible processing subsystem. There were 1,800 injected events in the specified RTL fault-injection campaign. 1,207 events were categorised as effectively mitigated, representing 67.1% successful mitigation, whereas 1,786 events were discovered by the reported combined architecture, representing 99.2% detection. Certain fault classes, such as specified double-bit memory faults and parity-protected detect-only events, are meant to be identified and reported rather than immediately fixed, which explains the distinction between detection and mitigation. Therefore, effective mitigation involves more than just physically restoring the damaged bit; it also involves masking, ECC correction, controlled reset recovery, and permitted safe-state responses. The findings corroborate the primary design concept used in this study, which states that various SoC resources should be protected according to how vital they are. While selective TMR enables robust protection to be focused on control-critical states, full-chip TMR would incur significant hardware and switching costs. While parity offers less expensive detection for less important registers, SEC-DED ECC offers a more appropriate approach for memory protection. For issues that cannot be resolved locally, centralised fault management and watchdog oversight provide system-level recovery. For coordinated low-power and radiation-tolerant SoC design, the study thus develops a VLSI-oriented architecture and RTL proof-of-concept. For the architecture under consideration, the provided findings demonstrate a balanced trade-off between power, FPGA resources, time, and fault response. It is not appropriate to interpret the quantitative findings as physical radiation probabilities or general cross-platform performance promises since they are still particular to the described FPGA-oriented dataset and RTL fault-injection campaign. The current work may be extended towards deployment-oriented radiation-tolerant devices with further hardware validation and physical radiation testing.

REFERENCES

- [1] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, Apr. 1992, doi: 10.1109/4.126534.
- [2] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *IEEE Transactions on Nuclear Science*, vol. 50, no. 3, pp. 583-602, Jun. 2003, doi: 10.1109/TNS.2003.813129.
- [3] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200-209, Apr. 1962, doi: 10.1147/rd.62.0200.
- [4] RISC-V International, "The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, RV32I Base Integer Instruction Set, Version 2.1," RISC-V Ratified Specifications Library, 2024.
- [5] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75-82, Apr. 1997, doi: 10.1109/2.585157.
- [6] AMD, "UltraFast Design Methodology Guide for FPGAs and SoCs (UG949): Gating the Clock Buffer," AMD Adaptive Computing Documentation, 2023.1, 2023.
- [7] AMD, "Vivado Design Suite User Guide: Power Analysis and Optimization (UG907)," AMD Adaptive Computing Documentation, 2026.1, 2026.
- [8] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305-316, Sep. 2005, doi: 10.1109/TDMR.2005.853449.
- [9] AMD, "Device Reliability Report (UG116), Rev. 10.20: SEU and Soft Error Rate Measurements," AMD Adaptive Computing Documentation, Jul. 2025.
- [10] AMD, "Soft Error Mitigation Controller v4.1 LogiCORE IP Product Guide (PG036)," AMD Adaptive Computing Documentation, Nov. 2023.