Chapter

Design of Low-Cost Reliable and Fault-Tolerant 32-Bit One Instruction Core for Multi-Core Systems

Shashikiran Venkatesha and Ranjani Parthasarathi

Abstract

Billions of transistors on a chip have led to integration of many cores leading to many challenges such as increased power dissipation, thermal dissipation, occurrence of faults in the circuits, and reliability issues. Existing approaches explore the usage of redundancy-based solutions for fault tolerance at core level, thread level, micro-architectural level, and software level. Core-level techniques improve the lifetime reliability of multi-core systems with asymmetric cores (large and small cores), which have gained momentum and focus among a large number of researchers. Based on the above implications, multi-core system using one instruction cores (MCS-OIC) factoring its features are proposed in this chapter. The MCS-OIC is an asymmetric multi-core architecture with MIPS core as the conventional core and OICs as the warm standby-redundant core. OIC executes only one instruction named 'subleq _ subtract if less than or equal to zero'. When there is one of the functional units (i.e., ALU) of any conventional core fails, the opcode of the instruction is sent to the OIC. The OIC decodes the instruction opcode and emulates the faulty instruction by repeated execution of the 'subleq' instruction, thus providing fault tolerance. To evaluate the idea, the OIC is synthesized using ASIC and FPGA. Performance implications due to OICs at instruction and application level are evaluated. Yield analysis is estimated for various configurations of multi-core system using OICs.

Keywords: fault tolerance, reliability, one instruction core, multi-core, yield

1. Introduction

Researchers have predicted about an eight percent increase in soft-error rate per logic state bit in each technology generation [1]. According to the International Telecommunication Roadmap for Semiconductors (ITRS) 2005 and 2011, reduction in dynamic power, increase in resilience to faults and heterogeneity in computing architecture pose a challenge for researchers. According to the International Roadmap for

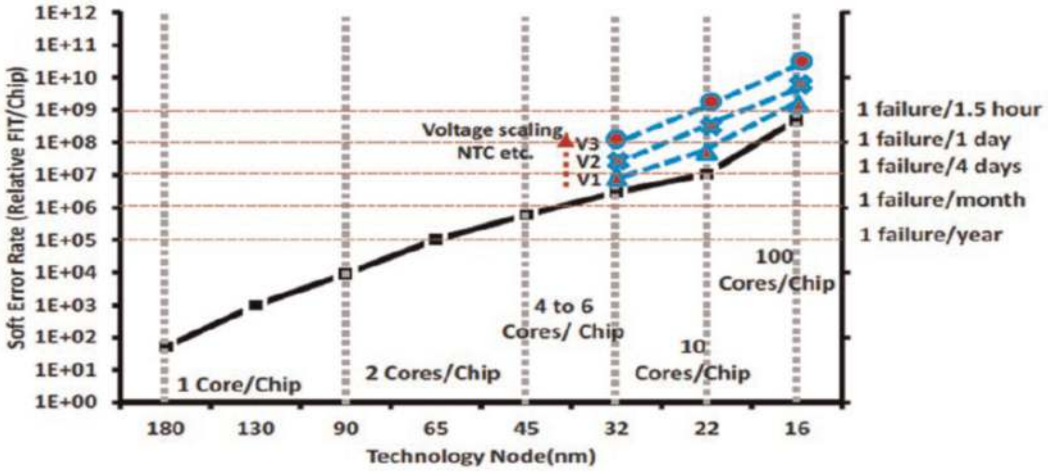


Figure 1.
SERs at various technology node.

Device and System (IRDS) roadmap 2017, device scaling will touch the physical limits with failures reaching one failure per hour as shown in **Figure 1**. The soft error rate (SER) is the rate at which a device or system encounters or is predicted to encounter soft errors per unit of time, and is typically expressed as failures-in-time (FIT). It can be seen, from **Figure 1** [2–4] that, at 16 nm process node size, a chip with 100 cores could come across one failure every hour due to soft errors.

This decrease in process node size and increase in integration density as seen in **Figure 1**, has the following effects.

1. Number of cores per chip has increased. Due to increase in number of cores, size of the last level cache (LLC) has increased. For example, NVIDIA's GT200 architecture GPU did not have an L2 cache, the Fermi GPU, Kepler GPU, Maxwell GPU has 768KB LLC, 1536KB LLC and 2048KB LLC respectively [5]. Similarly, Intel's 22 nm Ivytown processor has a 37.5 MB static random-access memory (SRAM) LLC (Rusu 2014) [6] and 32 nm Itanium processor had a 32 MB SRAM LLC (Zyuban 2013) [7]. Consequence of larger cache size has led to exponential increase in SER.

2. Low swing interconnect circuits are being used in CMOS transmission system. This has proved to be an energy efficient signalling system compared to conventional full swing interconnects circuits. However, incorrect sampling of the signals in low swing interconnect circuits together with interference and noise sources can induce transient voltages in wires or internal receiver nodes resulting in incorrect value being stored at receiver output latch [8].

This scenario can be envisaged as a "fault wall". In order to surmount the fault wall scenario, reliability has been identified as a primary parameter for future multi-core processor design [9, 10]. Similarly, ITRS 2005 and 2011, have also identified increase in resilience to faults as a major challenge for researchers. Hence, a number of researchers have started focusing on resilience to faults and reliability enhancement in multi-core processors. The chapter focuses on providing fault tolerance solutions for processor cores in multi-core systems.

2. Motivation

As seen in **Figure 1**, the total FIT per chip increases with number of cores per chip increasing. In order to accommodate higher number of cores per chip, (1) total FIT per chip has to be maintained constant (or no change), and (2) SER per core needs to be reduced. In the present-day processor cores, the frontend of the core comprises of decode queue, instruction translation lookaside buffer, and latches. The backend of the core comprises of arithmetic logic unit, register files, data translation lookaside buffer, reorder buffers, memory order buffer, and issue queue. SER from backend and the frontend of the core is 74.48% and 25.22% respectively. In the present processor cores, latches are hardened [11, 12] cache and large memory arrays are protected using error correcting codes (ECC) [13, 14]. The SER from backend of the processor is more when compared to front end and is mainly due to arithmetic logic unit. The FIT from the arithmetic logic unit of the processor core has started reaching higher levels which needs robust fault mitigation approaches for present and future processors. Hence addressing the reliability issues of the core (arithmetic logic unit in backend) is more significant in improving the reliability of the multi-core system [15, 16]. Conventional approaches to handle soft errors consumes more power and area. Hence, the chapter focuses on using heterogeneous model with low cost ("low cost" denote low power and lesser area of OICs) fault tolerant cores to improve reliability of multi-core systems.

2.1 Chapter contributions

Contributions of the chapter are briefly presented below.

1. The microarchitecture consisting of control and data path for OIC is designed. Four modes of operation in 32-bit OIC namely (a) baseline mode (b) DMR mode (c) TMR mode and (d) TMR with self-checking subtractor (TMR + SCS) are introduced.

2. The microarchitecture of 32-bit OIC and multi-core system integrated with 32-bit OIC are implemented using Verilog HDL. The design is synthesized in Cadence Encounter (R) RTL Compiler RC14.28 –V14.20 (Cadence design systems 2004) using TSMC 90nm technology library (tcbn90lphptc 150).

3. Dynamic power, area, critical path and leakage power for four modes of OIC are estimated and compared.

4. Dynamic power and area of OIC and URISC++ are compared.

5. Area and power are estimated for multi-core system consisting of 32-bit OIC.

6. The OIC is synthesized using Quartus prime Cyclone IVE (Intel, Santa Clara, CA) with device EP4CE115FE29C7. Number of logical elements and registers are estimated.

7. Number of logical elements and registers in OIC and URISC++ are compared.

8. Using Weibull distribution, the reliability for the four modes of OIC are evaluated and compared.

9. Using Weibull distribution, the reliability for OIC and URISC++ are evaluated and compared.

10. Performance overhead at instruction level and application level is estimated.

11. Yield analysis for proposed multi-core system with OICs is presented.

2.2 Chapter organization

The remaining portion of the chapter is organized as follows as: Section titled "3. An Overview on 32-bit OIC" presents (a) an outline of 32-bit OIC (b) one instruction set of OIC (c) modes of operation of OIC (d) microarchitecture of OIC (e) microarchitecture of multi-core system consisting of OIC (f) instruction execution flow in multi-core system using one instruction cores (MCS-OIC); Section titled "4. Experimental results and discussion" presents power, area, register and logical elements estimation for OIC, and power, area estimation for MCS-OIC; Section titled "5. Performance implications in multi-core systems" presents performance implications at instruction level and application level; Section titled "6. Yield analysis for MCS-OIC" presents yield estimates for the proposed MCS-OIC; Section titled "7. Reliability analysis of 32-bit OIC" presents reliability modelling of OIC and its estimate in different operational modes; the conclusion of the chapter is presented in the Section titled "8. Conclusion"; the relevant references are citated in the Section titled "References".

3. An overview on 32-bit one instruction core

A 32-bit OIC [17] is designed to provide fault tolerance to a multi-core system with 32-bit integer instructions of conventional MIPS cores. OIC is an integer processor. The terms "32-bit OIC" and "OIC" are interchangeably used in this thesis. OIC executes only one instruction, namely, "subleq – subtract if less than or equal". The OIC has three conventional subtractors and an additional self-checking subtractor. A conventional core that detects faults in one of the functional units (i.e., ALU) sends the opcode with operands to the OIC. In this thesis, the OIC is designed to support the instruction set of 32-bit MIPS core. However, it can be designed to support 32 bit ×86/ARM instruction set by making necessary changes in the instruction decoder. The OIC emulates the instruction by repetitively executing the subleq instruction in a predetermined manner. There are four modes of operation in OIC and they are (a) baseline mode (b) DMR mode (c) TMR mode and (d) TMR + Self Checking Subtractor (SCS) or TMR + SCS mode. TMR + SCS is the "high resilience mode" of OIC. Baseline mode is invoked only when soft error detection and correction alone are required.

3.1 One instruction set

"Subleq – subtract if less than or equal" is the only instruction executed by the OIC. The syntactic construct of the subleq instruction is given below.

Subleq A, B, C; Mem [B] = Mem [B] – Mem [A]

ADD a,a,b	**INC a**	**MOV a, b**	RSB b, a, b
1.Subleq a, z,2	1.Subleq One, z,2	1.Subleq a, a,2	1.Subleq a, b,2
2.Subleq z, b,3	2.Subleq z, a,3	2.Subleq b, z,3	2 ret
3 Subleq z, z,4	3.Subleq z, z,4	3.Subleq z, a,4	**DEC a**
4 ret	4.ret	4.Subleq z, z,5	Subleq one, a
		5.ret	ret

Table 1.
Sequence of synthesized Subleq instruction.

; If (Mem [B] ≤ 0) go to C;

It is interpreted as: "subtract the value at the memory location A from the value at the memory location B; store the result at the memory location B; If the value at the memory location B is less than or equal to zero, then jump to C." The subleq instruction is Turing complete. The instruction set of a core or processor is said to be Turing complete, if in principle, it can perform any calculation that any other programmable computer can. As an illustration, the equivalent synthesized subleq instructions for ADD, INC, MOV, DEC and RSB (Reverse subtract) instructions are given in the **Table 1**.

3.2 Modes of operation

The OIC operates in four modes as mentioned above. They are (a) baseline mode (b) DMR mode (c) TMR mode and (d) TMR + Self Checking Subtractor (SCS) or TMR + SCS mode.

a. *Baseline mode*: In this mode, only the self-checking subtractor is operational. The results from the subtractor are verified by the self-checker. If the results differ, the subtraction operation is repeated to correct the transient faults. Transient faults are detected and corrected in this mode. If the results do not match again, a permanent fault is detected.

b. *DMR mode*: In this mode, only two subtractors are operational. The results of the two subtractors are compared using a comparator. If the results differ, the subtraction operation is repeated to correct the transient faults. The transient faults are detected and corrected in this mode. If one of the two subtractors fails, a permanent fault is detected, and the OIC switches to baseline mode.

c. *TMR mode*: In this mode, all three subtractors are operational. The results from the three subtractors are compared using three comparators. The voters check the results from the comparators and perform majority voting. To correct the transient faults, the operations are repeated. If anyone subtractor fails, the faulty subtractor is disabled. In this mode, results from the redundant subtractors are fed back on special interconnects to the inputs of the multiplexer. OIC then switches to DMR mode. It is assumed that two subtractors do not fail simultaneously. Occurrence of one permanent fault is detected and tolerated in this mode.

d. *TMR + SCS mode:* TMR + SCS mode is the initial mode of operation in OIC. In this mode, all three subtractors and SCS are operational. Both permanent and transient faults are detected and corrected. The results of three subtractors and SCS are compared using a comparator. If the results differ, then entire operation is repeated to correct the transient faults. If results continue to differ, then OIC switches to TMR mode.

3.3 Micro-architecture of OIC

The micro-architecture of the OIC is given in **Figure 2**. The micro-architecture of the OIC can be divided into two parts: the control unit and data-path unit. The control unit consists of a 12-bit program counter (PC), an instruction decoder, a 12-bit control word memory register and control word memory. The control memory is safeguarded by (12, 4) Hamming codes [18]. All single-bit errors are detected and corrected by Hamming codes. The data-path unit consists of four multiplexers, one demultiplexer, three subtractors, one self-checking subtractor (SCS), three comparators and one voter unit. Normally, the register files occupy a large die area in a core and are exposed to high energy particles. In the spheres of replication, the register files also have high access latency and power overhead due to their fortification from ECC. The OIC does not have large register files that are likely to propagate transient faults or soft errors to other subsystems. The OIC uses very few registers. Once the operands from faulty core are admitted, they are stored in the registers. The results computed by the subtractors are compared and fed back on a separate interconnect line to the respective multiplexers. The intermediate results are not stored in the registers.

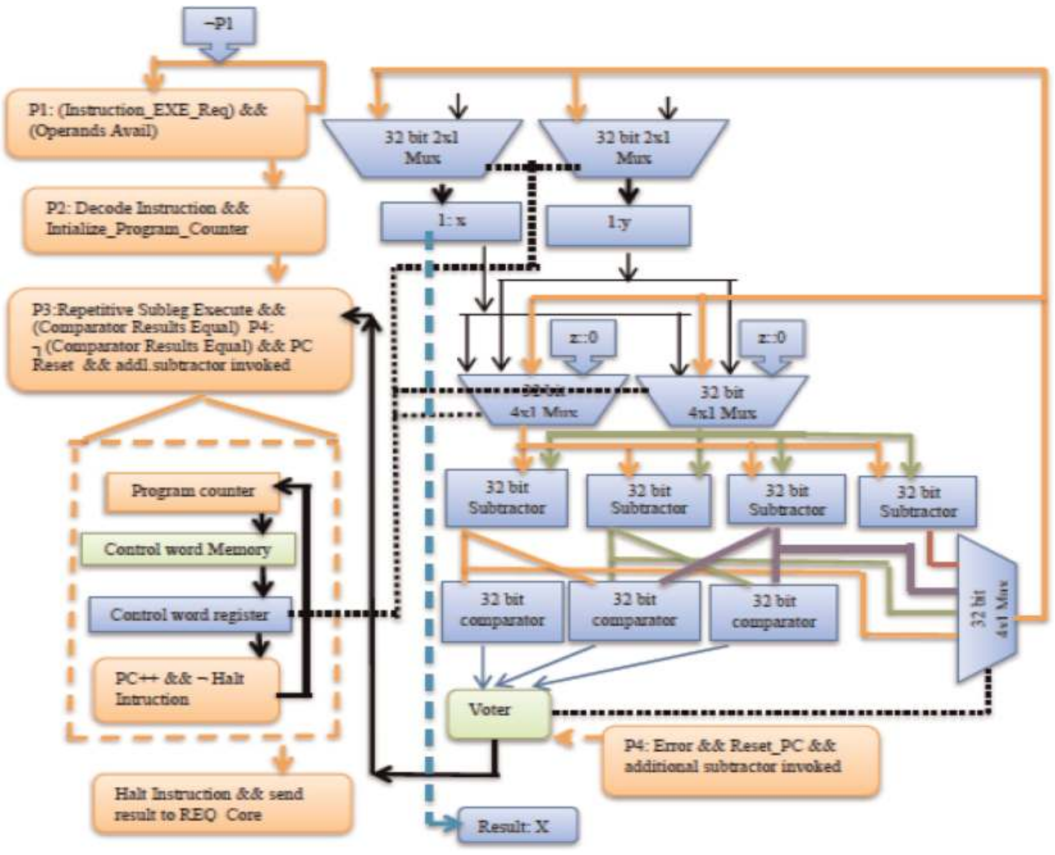

¬P1
P1: (Instruction_EXE_Req) && (Operands Avail)
P2: Decode Instruction && Intialize_Program_Counter
P3:Repetitive Subleq Execute && (Comparator Results Equal) P4: ¬(Comparator Results Equal) && PC Reset && addl.subtractor invoked
Program counter
Control word Memory
Control word register
PC++ && ¬Halt Intruction
Halt Instruction && send result to REQ Core
32 bit 2x1 Mux
32 bit 2x1 Mux
1:x
1:y
z::0
z::0
32 bit 4x1 Mux
32 bit 4x1 Mux
32 bit Subtractor
32 bit Subtractor
32 bit Subtractor
32 bit Subtractor
32 bit comparator
32 bit comparator
32 bit comparator
32 bit 4x1 Mux
Voter
P4: Error && Reset_PC && additional subtractor invoked
Result: X

Figure 2.
Control unit and data path unit of 32-bit OIC.

3.4 Microarchitecture and instruction execution flow in MCS-OIC

A Multicore system comprising one 32-bit MIPS core and one 32 bit OIC occupying the upper half and lower half portions respectively in the micro-architecture, is shown in **Figure 3**. The MIPS core is a five-stage pipelined scalar processor. Instruction Fetch (IF), Instruction Decode (ID), Execution (EXE), Memory access (MEM) and Write Back (WB) are the five stages in the MIPS pipeline. IF/ID, ID/EXE, EXE/MEM, and MEM/WB are the pipeline registers. PC is a program counter and LMD, Imm, A, B, IR, NPC, Aluoutput, and Cond are temporary registers that hold state values between clock cycles of one instruction. The fault detection logic (FDL) detects faults in all the arithmetic instructions (except logical instructions) by concurrently executing the instructions. The results of ID/EXE.Aluoutput and FDL are compared to detect the fault. If a fault is found then the pipeline is stalled. The IF/ID.opcode (in IR) and operands ID/EXE.A and ID/EXE.B are transferred to OIC as shown in **Figure 4**. The IF/ID.opcode is decoded and concurrently ID/EXE.A and ID/EXE.B values are loaded into the OIC registers (X & Y). The OIC.PC is initialized and simultaneously first control word from memory is loaded into its control word register. During every clock cycle, the control bits from control word register are sent to the selection lines of the multiplexer that control the input lines to the subtractors. At every clock cycle, subtraction is performed to emulate the instruction sent from the MIPS core. Finally,

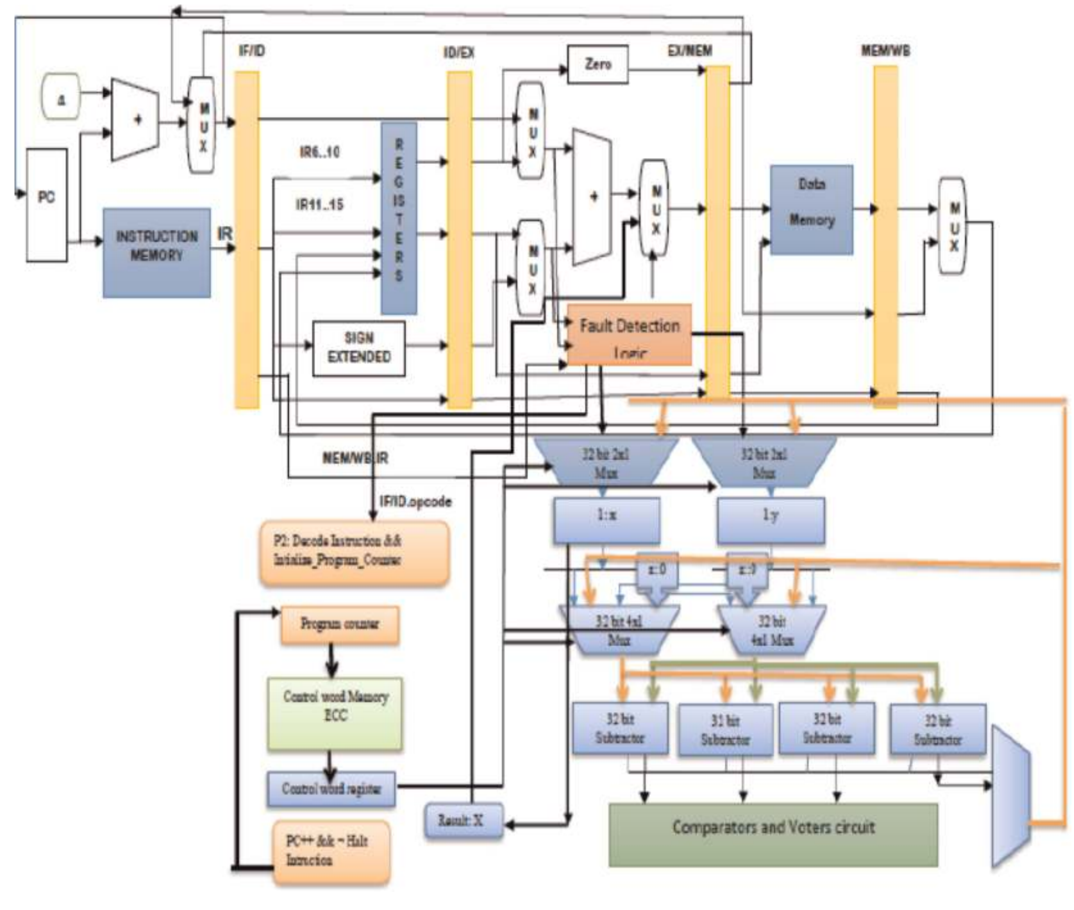


Figure 3.
Multi-core system consisting of one 32-bit MIPS core and one 32-bit OIC.

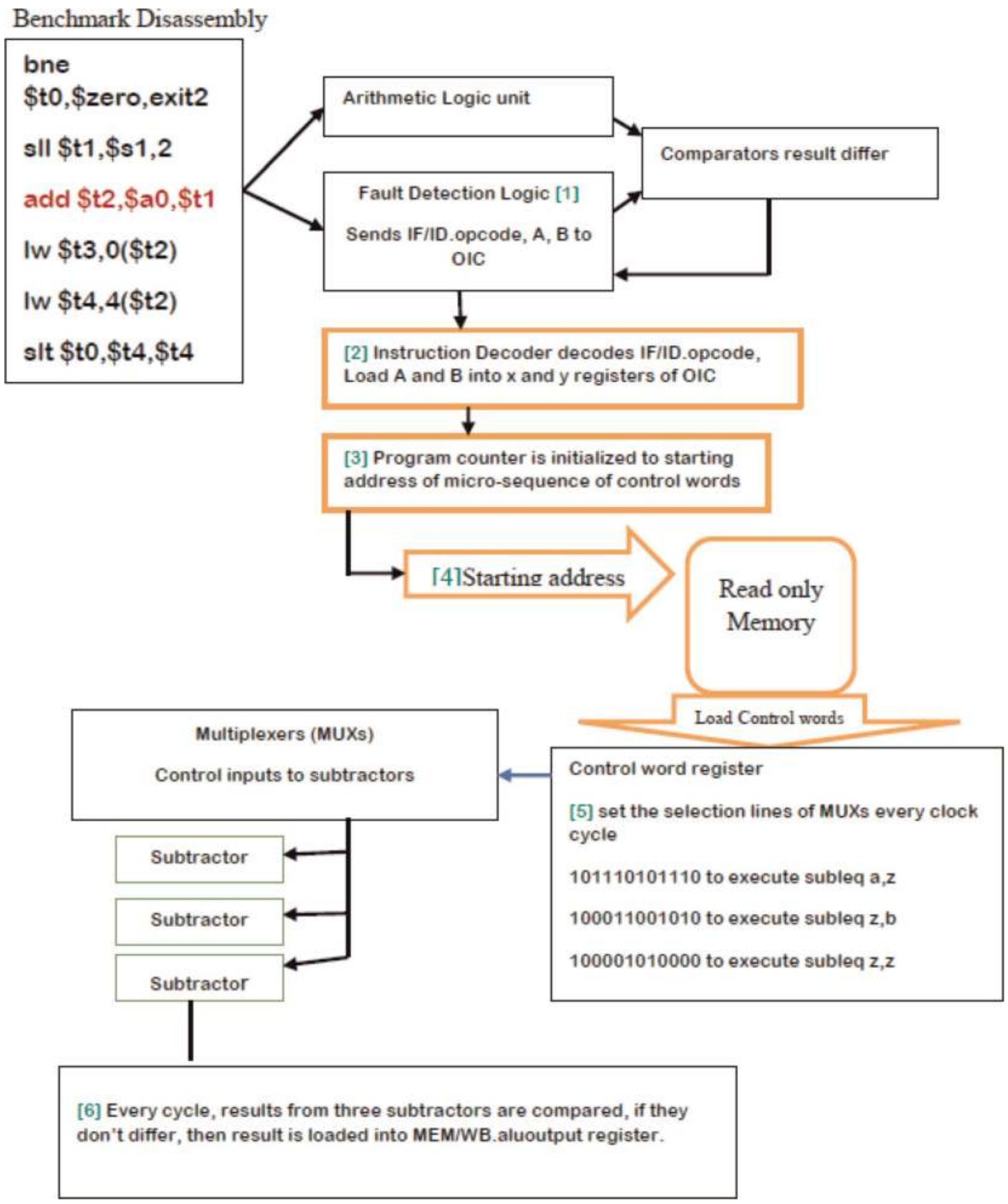

Benchmark Disassembly
bne $t0,$zero,exit2
sll $t1,$s1,2
add $t2,$a0,$t1
lw $t3,0($t2)
lw $t4,4($t2)
slt $t0,$t4,$t4
Arithmetic Logic unit
Comparators result differ
Fault Detection Logic [1]
Sends IF/ID.opcode, A, B to OIC
[2] Instruction Decoder decodes IF/ID.opcode, Load A and B into x and y registers of OIC
[3] Program counter is initialized to starting address of micro-sequence of control words
[4]Starting address
Read only Memory
Load Control words
Multiplexers (MUXs)
Control inputs to subtractors
Control word register
[5] set the selection lines of MUXs every clock cycle
101110101110 to execute subleq a,z
100011001010 to execute subleq z,b
100001010000 to execute subleq z,z
Subtractor
Subtractor
Subtractor
[6] Every cycle, results from three subtractors are compared, if they don't differ, then result is loaded into MEM/WB.aluoutput register.

Figure 4.
Sequence of events from fault detection to loading of results into Mem/WB.Aluoutput register of MIPS core.

the computed result is loaded into MEM/WB.Aluoutput and the MIPS pipeline operation is resumed. The sequence of events from fault detection to results loaded into MEM/WB.Aluoutput register of the MIPS core is shown in **Figure 4**.

4. Experimental results and discussion

The micro-architecture of the OIC is implemented using Verilog HDL and synthesised in ASIC and FPGA platforms to estimate hardware parameters (area, critical path delay, leakage power, dynamic power) and number of logical elements, register usage respectively. In the Section 4.1, comparison of area, power, registers and number of logical elements of OIC with an approach named URISC proposed by [19] and URISC ++ proposed by [20] is presented. Notably, URISC/URISC++ implement one instruction set. The URISC/URISC++, a co-processor for TigerMIPS, emulates instructions through the execution of subleq instruction. TigerMIPS performs static code insertion in both control flow and data flow invariants so as to detect faults by performing repeated executions of subleq within the co-processor. Comparative analysis on hardware parameters for different modes of OIC are discussed in Section 4.2.

ASIC simulation: The OIC given in **Figure 2** and multi-core system in **Figure 3** has been implemented using Verilog HDL and then synthesized in Cadence Encounter (R) RTL Compiler RC14.28 –V14.20 (Cadence design systems 2004) using TSMC 90 nm technology library (tcbn90lphptc 150). The area, power (dynamic, leakage, net, internal) and critical path delay are estimated for the OIC and tabulated in **Table 2**.

FPGA synthesis: The OIC is synthesized using Quartus prime Cyclone IVE with device EP4CE115FE29C7 and the results are illustrated in **Tables 3** and **4**.

Leakage power and dynamic power: Power dissipation shown in **Table 2** is understood as sum of dynamic power and static power (or cell leakage). Static power is consumed when gates are not switching. It is caused by current flowing through transistors when they are turned off and is proportional to the size of the circuit. Dynamic power is a sum of net switching power and cell internal power. The net switching power is the power dissipated in the interconnects and the gate capacitance.

Block name	Area (μm^2)	Leakage power (nW)	Internal (nW)	Net (nW)	Dynamic power (nW)	Critical path delay (ps)
Control path	590	39.87	79,498.48	21,881.40	10,1379.88	
(Control path + data path)	8122	704.08	10,51,631.88	346,487.45	13,98,115.34	8608
Sub blocks						
Subtractor	581	67.98	41,676.83	6711	48,387.83	
Comparator	615	67.04	42,457.83	9954.38	52,411.44	

Table 2.
Implementation of 32 bit OIC results using 90 nm TSMC technology.

(A) blocks	Logical elements	Dedicated registers
OIC (TMR + SCS)	530	160
Subtractor (1)	33	
Comparator (1)	43	
(B) modes	Logical elements	
Baseline	100	
DMR	303	
TMR	486	

Table 3.
FPGA synthesis results for OIC.

Cores	Logical elements	Dedicated registers
OIC	530	160
URISC	15,019	5232
URISC++	15,081	5233

Table 4.
FPGA synthesis results comparison.

The cell internal power is the power consumed within a cell by charging and discharging cell internal capacitances. The total power is a sum of the dynamic power and the leakage power.

Multi2sim (version 5.0): Multi2sim supports emulation for 32 bit MIPS/ARM binaries and simulation for 32-bit ×86 architectures. It performs both functional and timing simulations. The performance loss is estimated for compute intensive and memory intensive micro-benchmarks using a Multi2sim simulator. Performance loss for micro-benchmarks listed in **Table 6** are illustrated in **Figures 6–11**.

4.1 Comparative analysis: power, area, registers and logical elements

With the critical path delay at 8608 ps, the operating frequency of the circuit is 115 MHz with power supply at 1.2v. OIC is a low power core consuming 1.3 mW, with die area of 8122 μm^2. The die area of conventional MIPS core is 98,558 μm^2 which is 14.2× larger than OIC core. The MIPS core consumes a total power of 1.153 W and the 32-bit OIC consumes 1.39 mW; order of difference in powers of 10 is three. The registers in OIC are PC and temporary registers which hold the operands. But they are not designed and managed as a register file. **Tables 3** and **4** provide the register count and logical elements count for OIC and URISC++. The number of logical elements in OIC is 3.51% and 3.52% of the logical elements in URISC and URISC++ respectively. The number of registers in OIC is 3.05% of URISC++. URISC++ adds 62 logical elements and one additional register to the architecture of URISC. The logical elements in URISC++ consume 6.6 mW. URISC++ has 650 registers or 14.3% of registers in TigerMIPS. URISC++ has two large register files. URISC++ altogether consumes 1.96 W. Thus, OIC consumes less power than URISC++.

4.2 Comparative analysis: four modes of OIC

The critical path delay, area, dynamic power and leakage power for the four modes of OIC namely baseline mode, DMR mode, TMR mode and TMR + SCS mode are normalized to baseline mode and shown in **Figure 5**. The area overhead of TMR + SCS mode is 68.43% of the baseline, area overhead of TMR mode is 65.37% of the baseline and for DMR mode it is 51.4%. The comparators and subtractors occupy 22.71% and 28.6% of TMR + SCS mode area respectively. The size of the voter is negligible in TMR + SCS mode and TMR mode. In the critical path delay, 10% increase is noticed from the baseline to TMR + SCS mode. The critical path traverses from the subtractor input to the comparator, and then to the voter, passing through select logic and ends at an input line. Delay would not differ much between TMR mode and TMR + SCS mode.

Both the dynamic power and leakage power for TMR mode and DMR mode increase significantly due to redundant subtractors and comparators which are not in the baseline. The dynamic power overhead of TMR mode and DMR mode is 60% and

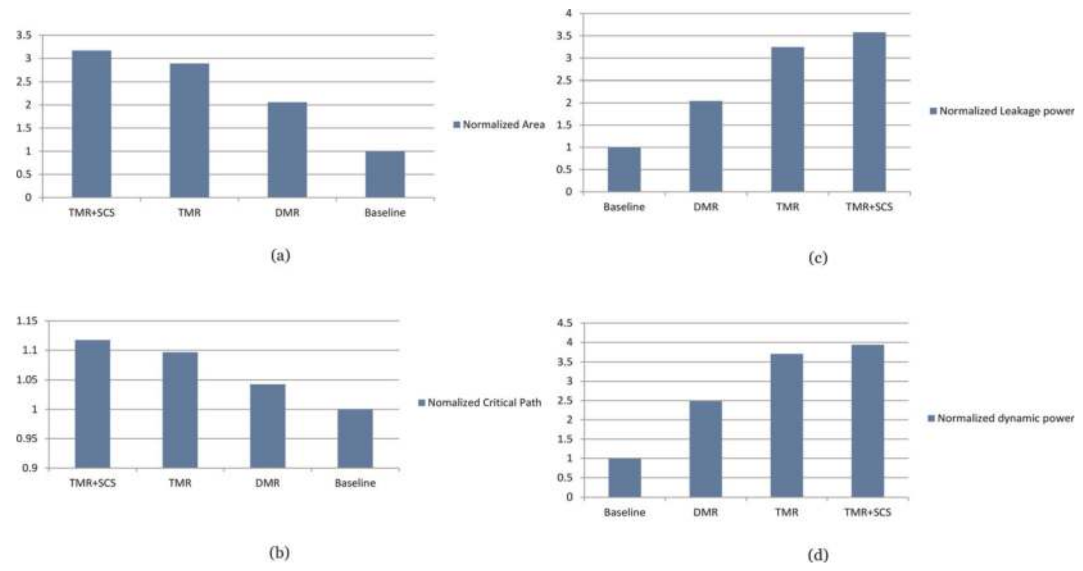


Figure 5.
(a) Area, (b) critical path delay, (c) leakage power and (d) dynamic power (y-axis—normalized values to baseline).

73% of the baseline. It is 75% for TMR + SCS mode. The static power or leakage power is proportional to the size of the circuit. The TMR + SCS mode has leakage power which is 76% more than the baseline. The TMR and DMR mode have leakage power which is 72% and 50% more than the baseline. In **Table 4** which depicts FPGA synthesis results, it is observed that the number of logical elements in TMR + SCS mode and DMR mode is 79% and 66% more than the baseline. From **Tables 2** and **4**, it is observed that TMR mode with additional self-checking subtractor in TMR + SCS mode costs more than the baseline, but still TMR + SCS/OIC will be a suitable fault tolerant core for a low power embedded system.

4.3 Power and area estimation for MCS-OIC

The area and power for the micro-architecture of multi-core system (one MIPS core with one OIC) shown in **Figure 3**, are estimated using ASIC simulation. The multi-core system occupies a total area of 306,283 μm^2 and consumes a total power of 1.1554 W. The FDL occupies an area of 6203 μm^2 which is 2% of the total area occupied by the system. The OIC occupies an area of 8122 μm^2 which is 2.6% of the total area occupied by the system. The FDL consumes a power of 1.2 mW and OIC consumes a power of 1.4 mW which are negligible when compared to the total power. The redundancy-based core level fault mitigation techniques/approaches such as Slipstream [21], dynamic core coupling (DCC) approach proposed by [22], configurable isolation [23], reunion is a fingerprinting technique proposed by Smolens et al. [24] have nearly 100% area overhead and obviously larger power overhead.

5. Performance implications in MCS-OIC

For every instruction emulated on OIC, an additional three clock cycles are needed for transfer of opcodes and operands, and two clock cycles are needed to resume the pipeline in the MIPS processor. The two terms defined below highlight the latency that incur in instruction execution, presented in the following subsection.

5.1 Performance overhead at instruction level

Definitions: (a) The **instruction execution time by emulation (IETE)** is defined as the number of cycles needed to execute the instruction on OIC. (b) **Total execution time (TET)** is defined as the sum of IETE and time (in clock cycles) to transfer opcodes, operands (from MIPS to OIC) and results (from OIC to MIPS). In other words, this indicates that time in clock cycles between pipeline stall and resumption of pipeline. The TET and IETE for instructions are tabulated in **Table 5**.

5.2 Performance overhead at application level

In the previous section, performance loss at instruction level caused due to transfer of operands and results back to host core is discussed. This would cause an accumulative loss in performance of application and the same is discussed in this section. The OIC supports 32 bit ISA of MIPS R3000/4000 processor operating at a frequency of 250 MHz. OIC operates at a frequency of 115 MHz, thereby incurring a performance loss while emulating the instructions from a faulty functional unit in MIPS core. The Multi2sim, a cycle accurate simulator together with a cross compiler, *mips-linux-gnu-gcc/mips-unknown-gnu-gcc* is used to estimate the simulation execution time for a set of micro-benchmarks. The emulation of only arithmetic instructions on OIC is considered for estimating the performance loss as they constitute nearly 60% of total instructions in integer application programs. The compute intensive and memory intensive micro-benchmark programs considered are listed in **Table 6**.

5.2.1. Memory intensive micro-benchmarks

The performance loss for memory intensive micro-benchmark programs, namely, binary search, quicksort (using recursion), and radix sort, are given in **Figures 6–8** respectively. The performance loss for CPU intensive micro-benchmark programs, namely, matrix multiplication, CPU scheduling, and sieve of Eratosthenes, are given in the **Figures 9–11** respectively. The baseline indicates the simulated execution time

Instructions	IETE	TET	Clock cycle in MIPS/LEON 2FT/3FT
ADD	4	9	1
MOV	5	10	1
INC	4	9	1
DEC	1	5	1
SUB	1	5	1
MUL	7 (per iteration)	Min 12	6
DIV	5 (per iteration)	Min 10	34

Table 5.
IETE and TET for instructions.

S. no	Micro-benchmark	CPU/memory intensive	Input form	Input size
1	Matrix multiplication (single/ multithreaded)	CPU	Matrix	[10 × 10], [100 × 100], [1000 × 1000] elements
2	Binary search (single/ multithreaded)	Memory	Array	3000, 30,000, 300,000 elements
3	Sieve of Eratosthenes	CPU	Array	1000, 10,000, 100,000 prime number limit
4	CPU-scheduling	CPU	Array	1000, 10,000, 100,000 processes
5	Quicksort (recursion)	Memory	Array	Sorted 100, 1000, 10,000 elements for worst case analysis
6	Radix sort	Memory	Array	1000, 10,000, 100,000 elements

Table 6.
CPU intensive and memory intensive micro-benchmarks.

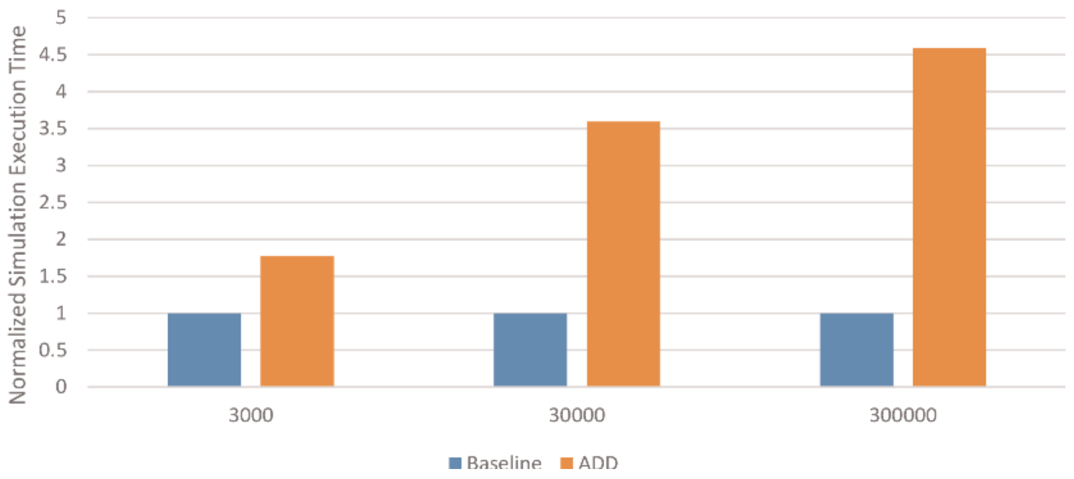


Figure 6.
Performance overhead in binary search by emulating ADD using subleq instruction.

of micro-benchmarks with no arithmetic instructions emulated on OIC. The performance loss is quantified for micro-benchmarks with respect to simulated execution time of the baseline (with varying input data sets/size).

As shown in **Figure 6**, Binary search with emulation of ADD instructions incurs performance loss of 1.77×, 3.59× and 4.59× with input size of 3000, 30,000 and 300,000 respectively, when compared to baselines. Significant proportion of ADD instructions is associated with incrementing or decrementing counters and effective addresses. OIC do not fetch operands or store results directly to main memory. Main memory latency is not taken into account for performance loss estimation. The number of ADD instructions executing as a part of the algorithmic phase of program execution does not increase exponentially with increase in input data sets. Hence, performance loss impact is minimal in algorithmic phase and is higher during fetching and storing of the input data sets. In case of multithreaded binary search, multi-core setup consisting of two cores core-0 and core-1 each with single thread is used to estimate the performance loss. The performance loss is similar to that of single

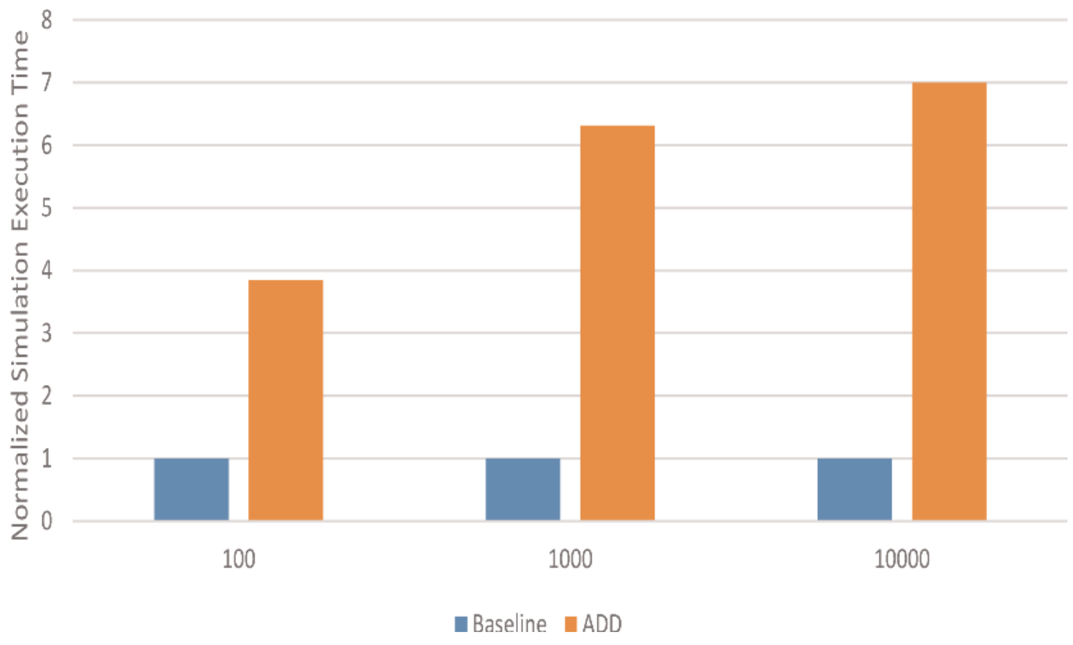


Figure 7.
Performance overhead in Quicksort by emulating ADD using subleq instruction.

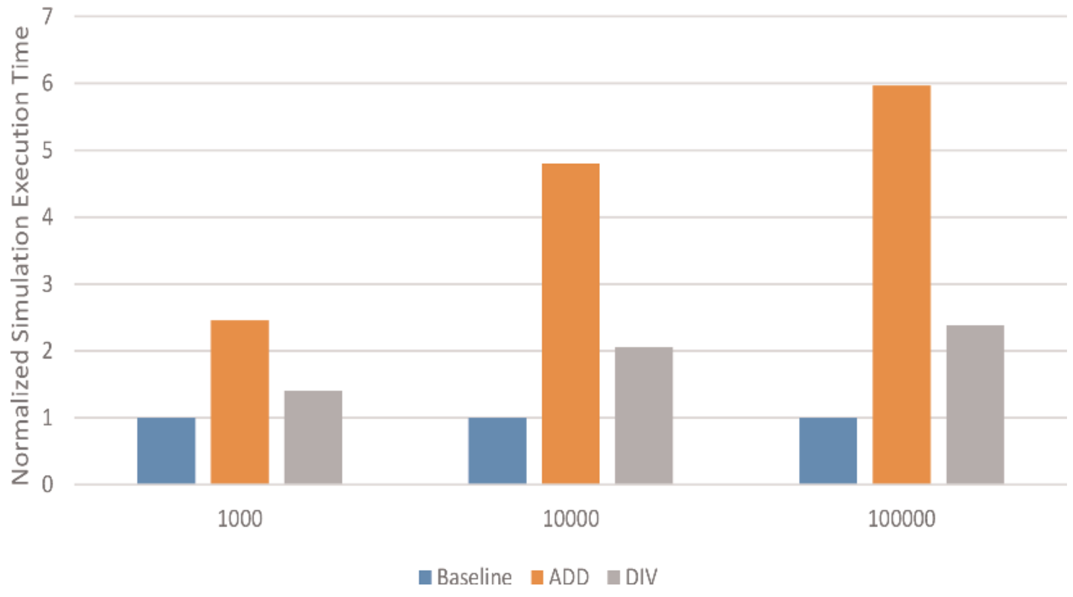


Figure 8.
Performance overhead in Radix sort by emulating ADD and DIV using subleq instruction.

threaded binary search. It is due to the fact that majority of the ADD instructions are associated with LOAD and STORE instructions.

Quicksort (with emulation of ADD instruction), implemented using recursion for sorted data elements (worst case analysis) incurs performance loss of 3.85×, 6.31×, and 6.99× for data size of 100, 1000 and 10,000 respectively as shown in **Figure** 7. For best case analysis of quicksort for 10,000 elements, performance loss reduces to 1.008×. Due to recursion, majority of ADD instructions are associated with LOAD/ STORE instructions. In radix sort, occurrence of ADD instructions is more compared to DIV instructions. Since it is memory intensive method of sorting, large number of

ADD instructions is used to increment counters and constants associated with LOAD/STORE instructions. The performance loss due to emulation of ADD instructions for radix sort is 2.45×, 4.79×, and 5.96× for input sizes of 1000, 10,000 and 10,000 as shown in **Figure 8**. For DIV instructions, performance loss is 1.4×, 2.05×, and 2.37× for input sizes of 1000, 10,000 and 10,000 elements.

As shown in **Figure 9**, matrix multiplication with emulation of ADD and MUL instructions executing in the algorithmic phase of the program, incurs a performance loss of <1.56×, 4.09×, 4.0×> (for ADD) and <1.632×, 7.62×, 7.99×> (for MUL), for input matrix sizes of 10 × 10, 100 × 100, and 1000 × 1000 respectively. In CPU scheduling, ADD and SUB instructions emulation incur a performance loss of <2.45×, 4.79×, 5.96×> and <1.4×, 2.05×, 2.3×> for input data set of 1000, 10,000 and 100,000 processes respectively as shown in **Figure 10**. In sieve of Eratosthenes, emulation of MUL and ADD instructions incur a performance loss of <1.89×, 5.03×, 7.63×> and <1.48×, 2.9×, 3.8×> for input data set size of 1000, 10,000 and 100,000 respectively as shown in **Figure 11**.

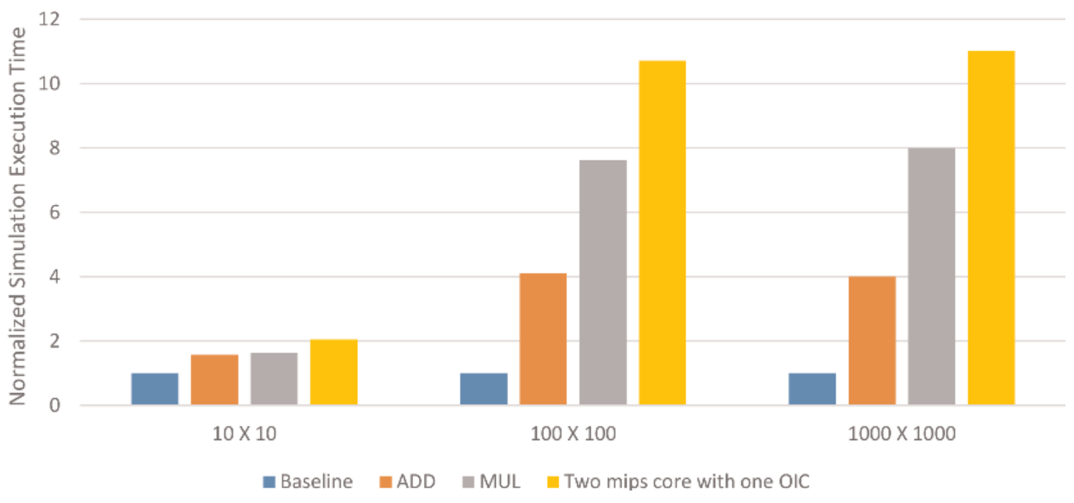


Figure 9.
Performance overhead in matrix multiplication by emulating ADD and MUL using subleq instruction.

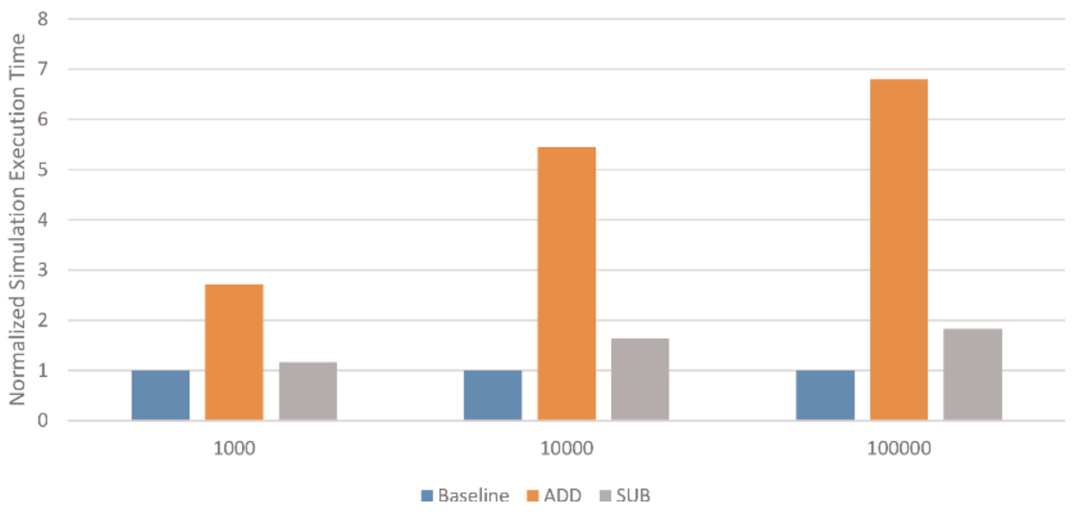


Figure 10.
Performance overhead in CPU scheduling by emulating ADD and SUB using subleq instruction.

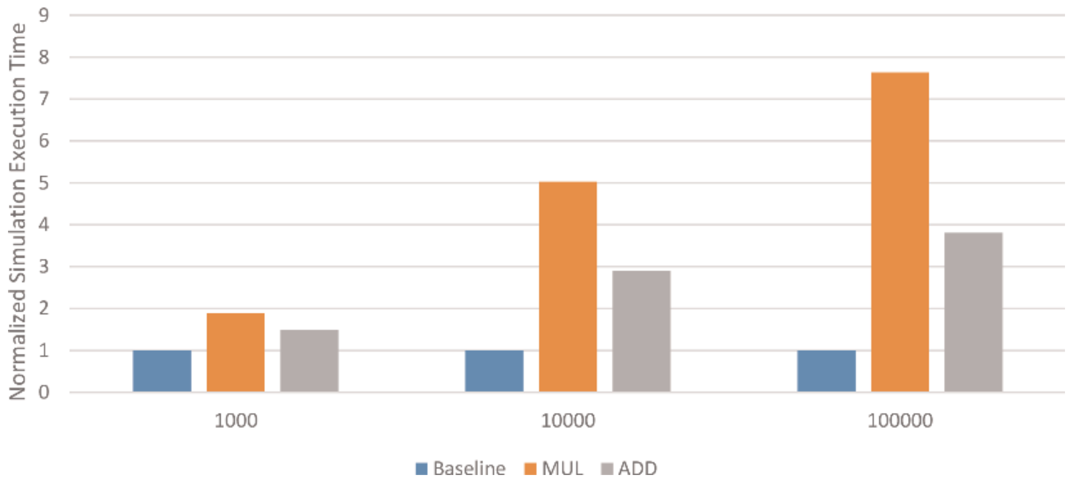


Figure 11.
Performance overhead in Sieve of Eratosthenes by emulating MUL and ADD using subleq instruction.

For multithreaded matrix multiplication, multi-core configuration consisting of two cores: core-0 and core-1 with single thread each is considered. The ADD and MUL instructions of core-0 and core-1 are emulated on single OIC due to failures in adder and multiplier units respectively. The performance loss is estimated as 2.04×, 10.07×, and 10.99× for matrix size of 10 × 10, 100 × 100, and 1000 × 1000 respectively as shown in **Figure 9**. Since, simultaneous access to single OIC from two cores is not permitted, performance loss includes the waiting time between subsequent ADD and MUL instructions emanating from core-0 and core-1. Waiting time alone is greater than 45% of the performance loss. In this multi-core configuration, consisting of two MIPS cores with single OIC, it bears the brunt of multiple functional unit failures in two cores. An Additional OIC would bring down the performance loss by 1.5× (for matrix size of 10 × 10) and 7× (for matrix size of 100 × 100/1000 × 1000) and eliminate the need for instructions to wait for execution on OIC. On 1:1 and 1: N basis i.e., one MIPS core with one or more OICs can scale to 100 MIPS core with 100 or more OICs.

It may be noticed that the performance loss does not vary when there is change of mode in OIC from TMR + SCR to TMR, or TMR to DMR as the number of instructions executed remains the same.

6. Yield analysis for MCS-OIC

This section examines the effect of fault tolerance provided in MCS-OIC on the yield. As discussed in the section which presents design of OIC, it is assumed that two subtractors do not fail simultaneously. In the TMR + SCR, TMR, and DMR modes, OIC repeats the instruction execution if the results differ, to avoid transient failures. The spatial and temporal redundancy to avoid permanent and transient faults in OIC makes it defect tolerant. The arithmetic logic unit in MIPS is protected by functional support provided by OIC. The remaining portion of MIPS are hardened and protected by ECC. The die yield for proposed different configurations of MCS-OIC is estimated using the equations presented below.

6.1 Terms and parameters

a. *Original die*: It is the die consisting of MIPS cores only.

b. *Fault tolerant die*: It is the die *consisting* of MIPS cores and OICs.

c. *Regular dies per wafer*: It is the number of original dies per wafer. The number of regular dies per wafer is estimated using the Eq. (1).

$$\text{Regular dies per wafer} = \frac{\pi(\text{diameter}/2)^2}{\text{Area}} - \frac{\pi \times \text{diameter}}{\sqrt{2} \times \text{Area}} \tag{1}$$

Where diameter refers to the diameter of the wafer, Area refers to the area of the die.

d. *Die yield*: Ignoring full wafer damage, the yield for single die is approximated using negative binomial approximation as given in the Eq. (2).

$$\textit{Die yield} = \left(1 + \frac{\text{defect density} \times \text{Area}}{\text{cp}}\right)^{-\text{cp}} \tag{2}$$

Where cp denotes cluster parameter or manufacturing complexity, defect density denotes number of defects per unit area.

e. *Regular working dies per wafer*: It is die yield times the regular dies per wafer. It is estimated using the Eq. (3).

$$\text{Regular working dies per wafer} = \left(1 + \frac{\text{defect density} \times \text{Area}}{\text{cp}}\right)^{-\text{cp}} \cdot \left(\frac{\pi(\text{diameter}/2)^2}{\text{Area}} - \frac{\pi \times \text{diameter}}{\sqrt{2} \times \text{Area}}\right) \tag{3}$$

f. *Regular fault tolerant dies per wafer*:

The area of fault tolerant core is expressed as summation of area of original die and area of OIC. If the area of OIC is expressed as $\delta(0 < \delta < 1)$ times the area of original design then $((1 + \delta) \times$ area of the original design)) denotes the area of the fault tolerant die. By substituting $(1 + \delta) \times$ area in the number of regular fault tolerant cores per wafer can be estimated and is given in the Eq. (4).

$$\text{Regular fault tolerant dies per wafer} = \frac{\pi(\text{diameter}/2)^2}{(1+\delta)\text{Area}} - \frac{\pi \times \text{diameter}}{\sqrt{(2\ (1+\delta)\ \text{Area})}} \tag{4}$$

g. *Regular working fault tolerant dies per wafer*: It is die yield times the regular fault tolerant die. It is estimated using the Eq. (5).

$$\text{Regular working fault tolerant dies per wafer} = \left(1+\frac{\text{defect density} \times \text{Area}}{\text{cp}}\right)^{-\text{cp}} \cdot \left(\frac{\pi(\text{diameter}/2)^2}{(1+\delta)\text{Area}} - \frac{\pi \times \text{diameter}}{\sqrt{(2(1+\delta)\text{Area})}}\right) \tag{5}$$

6.2 Parametric evaluation and discussion

The die yield for the original die and fault tolerant die estimated for one MIPS core with one/two/four/ OICs, two MIPS core with one/two/four/ OICs, four MIPS core with one/two/four/ OICs, and eight MIPS core with one/two/four/six OICs is tabulated in **Tables 7–10** respectively. The defect density is varied from 9.5, 5.0, to 1.0 and wafer diameters varied from 300 mm, 200 mm to 100 mm to estimate die yield of the original die and fault tolerant die. The cp is fixed at 4.0. The die yield of the original die at defect densities are 1.0, 5.0, and 9.5 are 0.9971, 0.9855, and 0.9727 respectively. The die yields for three fault tolerant dies each consisting of one MIPS core with first die with one OIC, second with two OICs, third with four OICs for 300 mm wafer with defect density at 1.0 is <0.9970/0.9969/0.9967> respectively as shown in the **Table 7**, which is slightly lesser than the yield of the original die. The average of the differences between yield of original die and fault tolerant dies with defect density 1.0 is 0.0002 which is negligible value. Similarly, the average of the differences between yield of original die and fault tolerant dies at defect density 5.0 and 9.5 are 0.0009 and 0.0017 respectively. It is observed that an increase in the defect density decreases the yield.

Wafer diameter		**100 mm**			**200 mm**			**300 mm**		
Defect density		9.5	5.0	1.0	9.5	5.0	1.0	9.5	5.0	1.0
Number of regular dies per wafers		26,489	26,489	26,489	10,6781	10,6781	10,6781	24,0876	24,0876	24,0876
Die yield of original die		0.9727	0.9855	0.9971	0.9727	0.9855	0.9971	0.9727	0.9855	0.9971
Number of regular working dies per wafer		25,767	26,106	26,412	10,3870	10,5237	10,6470	23,4309	23,7391	24,0174
Number of regular fault tolerant dies per wafer	One OIC	25,768	25,768	25,768	103,884	103,884	103,884	234,347	234,347	234,347
	Two OICs	25,084	25,084	25,084	101,139	101,139	101,139	228,163	228,163	228,163
	Four OICs	23,820	23,820	23,820	96,061	96,061	96,061	216,723	216,723	216,723
Die yield of fault tolerant die	One OIC	0.9719	0.9851	0.9970	0.9719	0.9851	0.9970	0.9719	0.9851	0.9970
	Two OICs	0.9712	0.9847	0.9969	0.9712	0.9847	0.9969	0.9712	0.9847	0.9969
	Four OICs	0.9697	0.9839	0.9967	0.9697	0.9839	0.9967	0.9697	0.9839	0.9967
Number of regular working fault dies per wafer	One OIC	25,046	25,385	25,691	10,0974	10,2340	103,573	227,784	230,864	233,645
	Two OICs	24,363	24,701	25,007	98,231	99,595	100,828	221,603	224,681	227,461
	Four OICs	23,100	23,437	23,743	93,157	94,518	95,750	210,170	213,243	216,021

Table 7.
Die yield for fault tolerant die consisting of one MIPS core with one/two/four OICs.

Wafer diameter		**100 mm**			**200 mm**			**300 mm**		
Defect density		9.5	5.0	1.0	9.5	5.0	1.0	9.5	5.0	1.0
Number of regular dies per wafers		13,159	13,159	13,159	53,220	53,220	53,220	120,182	120,182	120,182
Die yield for original die		0.9463	0.9713	0.9942	0.9463	0.9713	0.9942	0.9463	0.9713	0.9942
Number of regular working dies per wafer		12,454	12,782	13,082	50,368	51,694	52,911	113,740	116,736	119,484
Number of regular fault tolerant dies per wafer	One OIC	12,977	12,977	12,977	52,487	52,487	52,487	118,529	118,529	118,529
	Two OICs	12,494	12,494	12,494	50,544	50,544	50,544	114,150	114,150	114,150
	Four OICs	12,459	12,459	12,459	50,403	50,403	50,403	113,833	113,833	113,833
Die yield for fault tolerant die	One OIC	0.9456	0.9709	0.9941	0.9456	0.9709	0.9941	0.9456	0.9709	0.9941
	Two OICs	0.9449	0.9705	0.9940	0.9449	0.9705	0.9940	0.9449	0.9705	0.9940
	Four OICs	0.9428	0.9693	0.9937	0.9428	0.9693	0.9937	0.9428	0.9693	0.9937
Number of regular working fault tolerant dies per wafer	One OIC	12,272	12,600	12,900	49,636	50,962	52,177	112,091	115,085	117,830
	Two OICs	12,095	12,423	12,723	48,924	50,249	51,464	110,486	113,478	116,222
	Four OICs	11,592	11,919	12,219	46,900	48,222	49,435	105,923	108,908	111,650

Table 8.
Die yield for fault tolerant die consisting of two MIPS core with one/two/four OICs.

The die yield of the fault tolerant dies each consisting of two MIPS cores with < one/two/four> OICS with defect density 1.0 is <0.9941, 0.9940, 0.9937> respectively as shown in **Table 8**. The die yield of the original die at defect densities 1.0, 5.0, and 9.5 is 0.9942, 0.9713, and 0.9463 slightly higher than yield of fault tolerant dies. The average of the differences between yield of original die and fault tolerant dies is 0.00026. The average of the differences between yield of original die and fault tolerant dies increases by 0.0009 and 0.0018 for defect density 5.0 and 9.5 respectively.

The die yields of the original die at defect densities 1.0, 5.0, and 9.5 are 0.9884, 0.9436, and 0.8963 respectively. From **Table 9**, the die yield of the fault tolerant dies each consisting of four MIPS cores with <one/two/four> OICS with defect density 1.0 are <0.9883, 0.9882, 0.9880> respectively. It is observed that average of the differences between yield of the original die and fault tolerant dies at varying defect densities is similar with other alternatives discussed above.

From **Table 10**, the die yield of the fault tolerant dies each consisting of eight MIPS cores with <one/two/four/six> OICS with defect density 1.0 is <0.9769, 0.9767, 0.9765, 0.9764> respectively. The die yield of the original die at defect densities 1.0, 5.0, and 9.5 is 0.9769, 0.8912, and 0.8057 respectively. The average of the differences between the original die and fault tolerant dies with defect density of 9.5 is 0.0031, is the highest among the averages. From this data, it is inferred that larger chips with increasing redundancy widens gap between the yield of the original dies and fault

Wafer diameter		**100 mm**			**200 mm**			**300 mm**		
Defect density		9.5	5.0	1.0	9.5	5.0	1.0	9.5	5.0	1.0
Number of regular dies per wafers		6519	6519	6519	26,489	26,489	26,489	59,910	59,910	59,910
Die yield for original die		0.8963	0.9436	0.9984	0.8963	0.9436	0.9984	0.8963	0.9436	0.9984
Number of regular working dies per wafer		5843	6152	6444	23,744	24,997	26,182	53,700	56,536	59,216
Number of working fault tolerant dies per wafer	One OIC	6474	6474	6474	26,305	26,305	26,305	59,495	59,495	59,495
	Two OICs	6428	6428	6428	26,124	26,124	26,124	59,085	59,085	59,085
	Four OICs	6340	6340	6340	25,768	25,768	25,768	58,282	58,282	58,282
Die yield for fault tolerant die	One OIC	0.8956	0.9433	0.9883	0.8956	0.9433	0.9883	0.8956	0.9433	0.9883
	Two OICs	0.8949	0.9429	0.9882	0.8949	0.9429	0.9882	0.8949	0.9429	0.9882
	Four OICs	0.8929	0.9417	0.9880	0.8929	0.9417	0.9880	0.8929	0.9417	0.9880
Number of regular working fault tolerant dies per wafer	One OIC	5798	6106	6398	23,561	24,814	25,998	53,288	56,122	58,800
	Two OICs	5753	6062	6353	23,381	24,633	25,817	52,881	55,713	58,391
	Four OICs	5623	5930	6221	22,855	24,104	25,286	51,694	54,520	57,195

Table 9.
Die yield for fault tolerant die consisting of four MIPS core with one/two/four OICs.

Wafer diameter		**100 mm**			**200 mm**			**300 mm**		
Defect density		9.5	5.0	1.0	9.5	5.0	1.0	9.5	5.0	1.0
Number of regular dies per wafers		3217	3217	3217	13,159	13,159	13,159	29,827	29,827	29,827
Die yield for original die		0.8057	0.8912	0.9770	0.8057	0.8912	0.9770	0.8057	0.8912	0.9770
Number of regular working dies per wafer		2592	2867	3143	10,603	11,728	12,856	24,034	26,584	29,141
Number of regular fault tolerant dies per wafer	One OIC	3205	3205	3205	13,113	13,113	13,113	29,723	29,723	29,723
	Two OICs	3194	3194	3194	13,068	13,068	13,068	29,620	29,620	29,620
	Four OICs	3172	3172	3172	12,977	12,977	12,977	29,415	29,415	29,415
	Six OICs	3150	3150	3150	12,888	12,888	12,888	29,214	29,214	29,214
Die yield for fault tolerant die	One OIC	0.8051	0.8909	0.9769	0.8051	0.8909	0.9769	0.8051	0.8909	0.9769
	Two OICs	0.8040	0.8902	0.9767	0.8040	0.8902	0.9767	0.8040	0.8902	0.9767
	Four OICs	0.8028	0.8895	0.9765	0.8028	0.8895	0.9765	0.8028	0.8895	0.9765
	Six OICs	0.8016	0.8888	0.9764	0.8016	0.8888	0.9764	0.8016	0.8888	0.9764
Number of regular working fault tolerant dies per wafer	One OIC	2581	2856	3131	10,559	11,683	12,810	23,933	26,481	29,037
	Two OICs	2559	2833	3109	10,470	11,592	12,719	23,732	26,277	28,831
	Four OICs	2537	2811	3087	10,382	11,503	12,629	23,535	26,075	28,628
	Six OICs	2516	2790	3065	10,296	11,415	12,541	23,340	25,877	28,428

Table 10.
Die yield for fault tolerant die consisting of eight MIPS core with one/two/four/six OICs.

tolerant dies. Thus, a trade-off exists between the die yield and fault tolerance provided by the design alternatives (discussed above) having redundancy ranging between 2% and 11%.

7. Reliability analysis of 32-bit OIC

In order to assess the endurance for the four modes of OIC, reliability is evaluated and compared. The reliability, denoted by R(t), is defined as the probability of its survival at least until time t, which is estimated using Weibull distribution and can be determined in the following manner:

$$R(t) = P(T > t) = e^{-\lambda t^{\beta}} \tag{6}$$

where β is the shape parameter, T denotes the lifetime and λ denotes the failure rate of a component. Defect induced faults occur in the early stage of the lifetime, but the wear-out induced faults increase in the tail end of the lifetime. $\beta < 1$, is used to model infant mortality and it is a period of growing reliability and decreasing failure rate. When $\beta = 1$, the R(t) of Weibull distribution and exponential distribution are identical. $\beta > 1$, is used to model wear out and the end of useful life where failure rate is increasing. The initial failure rate is computed using the failure rate formula:

$$\lambda = (C_1\pi_T\pi_V + C_2\pi_E)\pi_Q\pi_L \tag{7}$$

here, C_1, C_2 are the complexity factors, $\pi_T, \pi_V, \pi_E, \pi_Q, \pi_L$ are temperature, voltage stress, environment, quality and learning factors respectively. Failure rate λ is assumed as a function of the number of logical elements in the micro-architectural components.

The reliabilities of the four modes of OIC given in the Eqs. (8)−(11) are expressed in terms of $R_{\text{select logic}}(t), R_{\text{sub}}(t), R_{\text{sub-sc}}(t), R_{\text{comp}}(t)R_{\text{voter}}$ which denote the reliabilities of select logic, subtractor, SCS, comparator and voters logic respectively.

TMR + SCS mode reliability is expressed as:

$$R_{\text{TMT+SCS}}(t) = R_{\text{sub}-\text{sc}}(t)R_{\text{select logic}}(t)R_{\text{comp}}(t)R_{\text{voter}}(t)\sum_{i=2}^{4}\binom{4}{i}(R_{\text{sub}}(t))^i(1-R_{\text{sub}}(t))^{4-i} \tag{8}$$

TMR mode reliability is expressed as:

$$R_{\text{TMR}}(t) = R_{\text{select logic}}(t)R_{\text{comp}}(t)R_{\text{voter}}(t)\sum_{i=2}^{3}\binom{3}{i}(R_{\text{sub}}(t))^i(1-R_{\text{sub}}(t))^{3-i} \tag{9}$$

DMR mode reliability is expressed as:

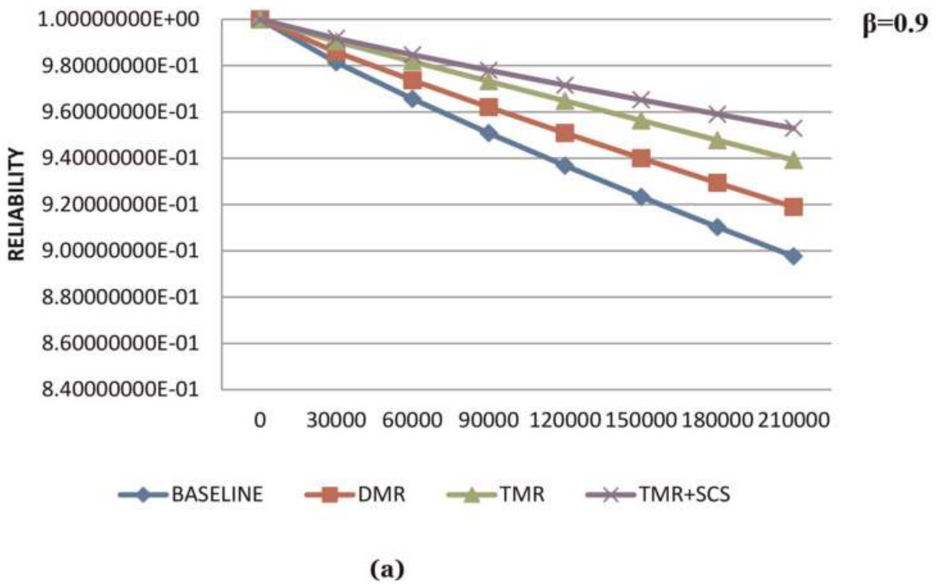
$$R_{\text{DMR}}(t) = R_{\text{select logic}}(t)R_{\text{comp}}(t)\sum_{i=1}^{2}\binom{2}{i}(R_{\text{sub}}(t))^i(1-R_{\text{sub}}(t))^{2-i} \tag{10}$$

Baseline mode reliability is expressed as:

$$R_{\text{baseline}}(t) = R_{\text{select logic}}(t)R_{\text{sub-sc}}(t) \tag{11}$$

The reliabilities are plotted for TMR + SCS, TMR, DMR and Baseline modes in **Figure 12** for β = 0.9 and 1.0, (which denote defect induced fault phase) and in **Figure 13** for β = 1.1 and 1.2 (which denote wear out induced fault phase). λ is a function of number of logical elements as given in the **Table 3**.

In all these cases, TMR + SCS mode is observed to have a better failure tolerance when compared to all other modes. For β = 1.2, the reliabilities of TMR mode and DMR mode are less than that of TMR + SCS mode during the interval 3×10^4 to 15×10^4 hours, as illustrated in **Figure 13**. The levels of reliability of TMR modes decline far below DMR, and baseline modes in wear out induced fault phase due to the



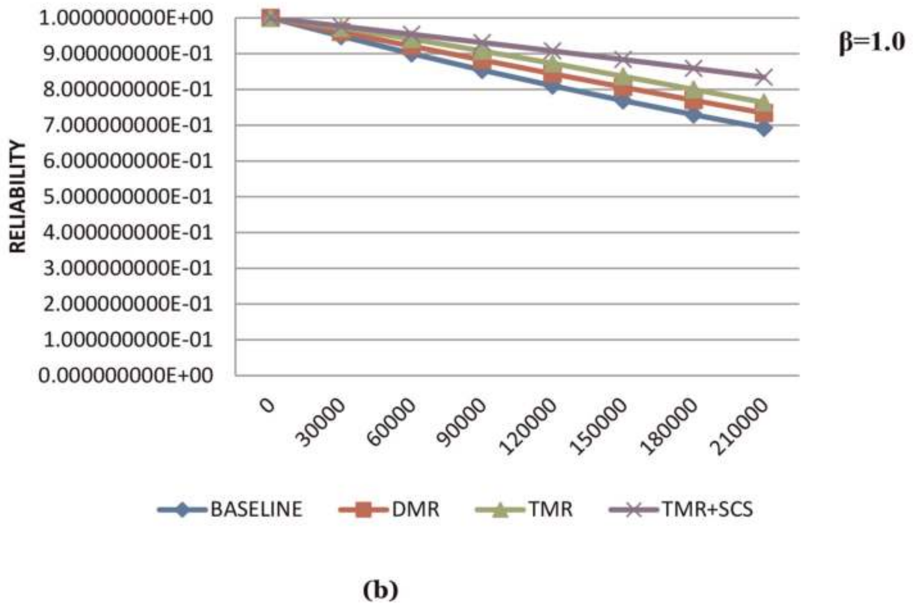
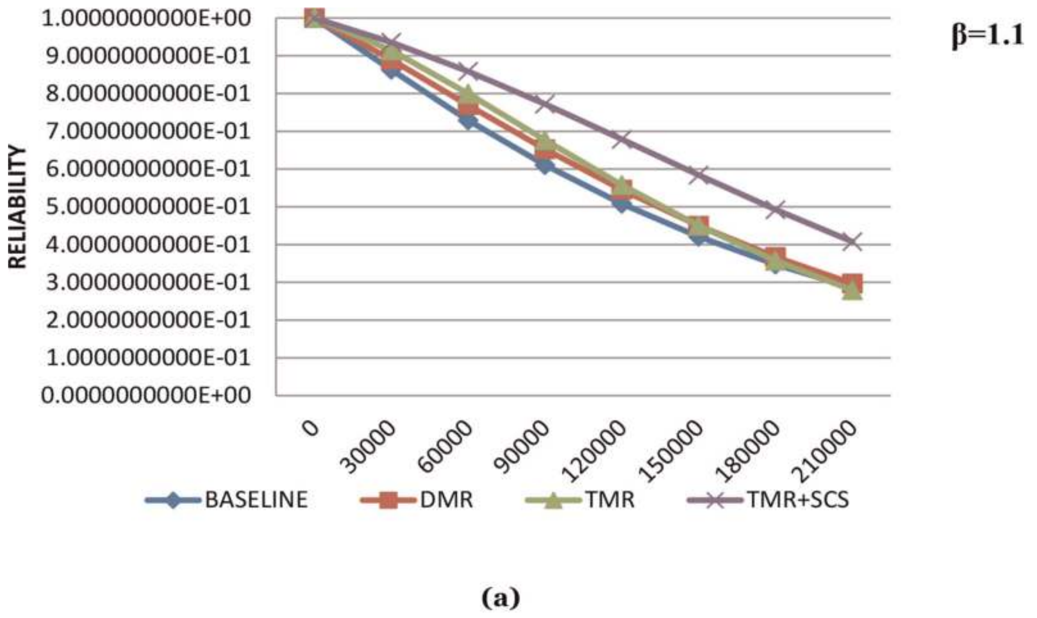


Figure 12.
Reliability vs. time for (a) β = 0.9 and (b) β = 1.0.



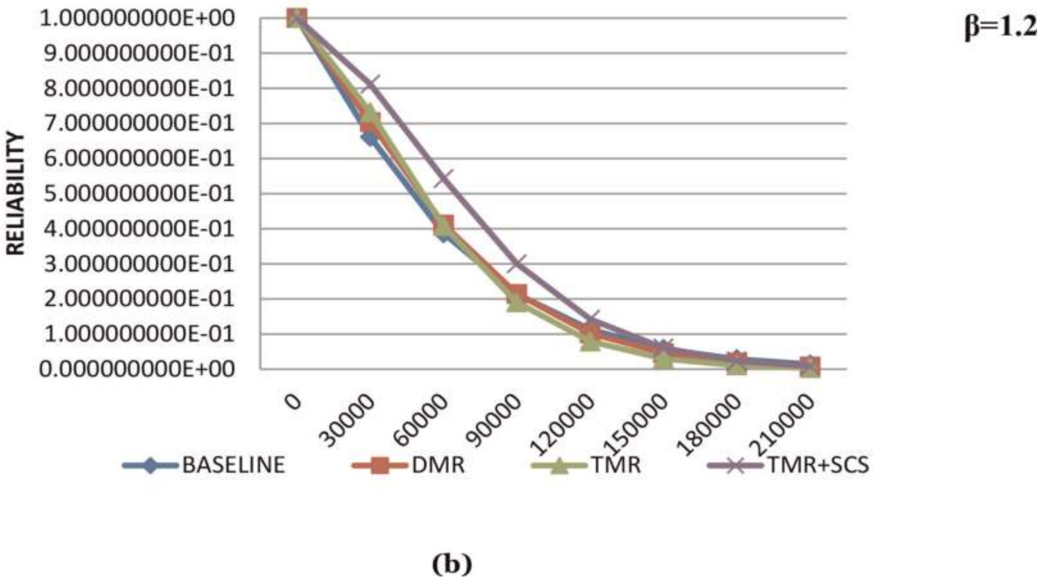


Figure 13.
Reliability vs. time for (a) β = 1.1 and (b) β = 1.2

fact that a single component reliability is below 0.5 and that the redundancy does not have any merit in the TMR mode. In **Table 11**, reliability of subtractor goes below 0.5 at t = 180,000 h or 20.5 years and reliability gap between TMR and DMR widens endorses the above argument.

7.1 Comparative analysis: OIC and URISC/URISC++

In this section, reliability of OIC is compared with that of URISC++. The reliability function of Weibull distribution with λ as a function of number of logical elements is used to estimate the reliability of URISC/URISC++. The number of logical elements in

t (h)	*R* (subtractor)	*R* (comparator)	*R* (TMR)	*R* (DMR)
120,000 (13.7 years)	0.6256	0.6948	0.16931	0.2112
150,000 (17.12 years)	0.5417	0.6226	0.09060	0.1272
180,000 (20.5 years)	0.4663	0.5545	0.04633	0.07706

Table 11.
Reliabilities of components in OIC for β = 1.2.

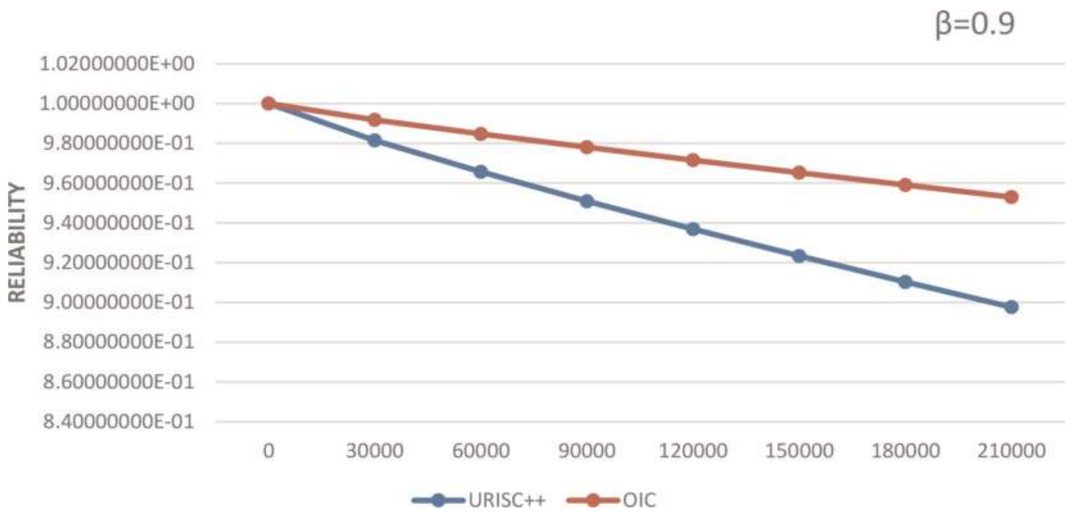


Figure 14.
β = *0.9 reliability vs. time (hours).*

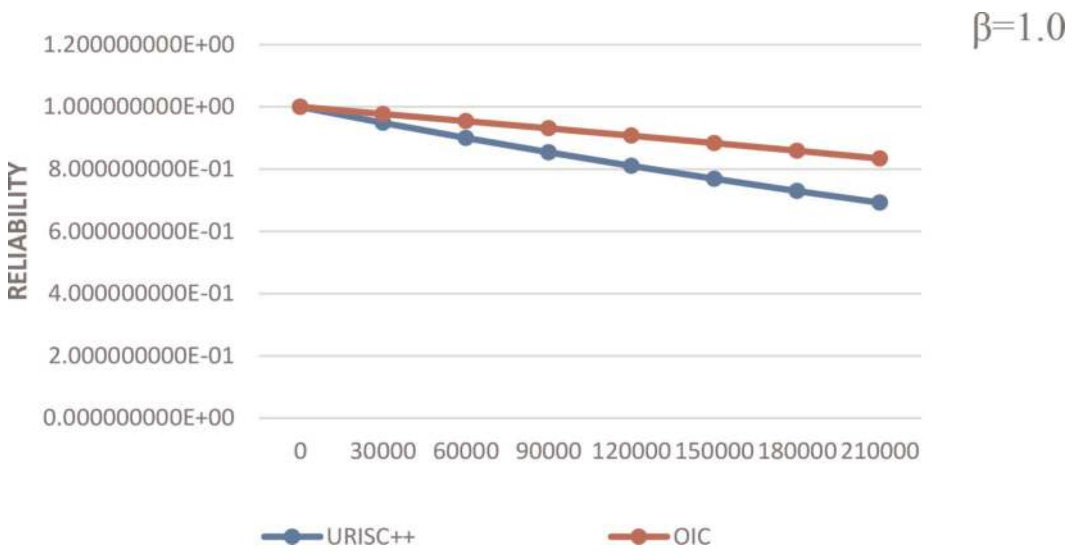


Figure 15.
β = *1.0 reliability vs. time (hours).*

OIC and URISC++ are given in **Table 4**. In the defect induced fault phase (β = 0.9 and β = 1.0), a drastic fall in the URISC++ reliability is observed as shown in **Figures 14** and **15.** OIC continues to maintain a reliability of 0.96, unlike URISC++ with endurance reaching 0.87 after 210,000 hours. In the wear-out induced fault phase, the

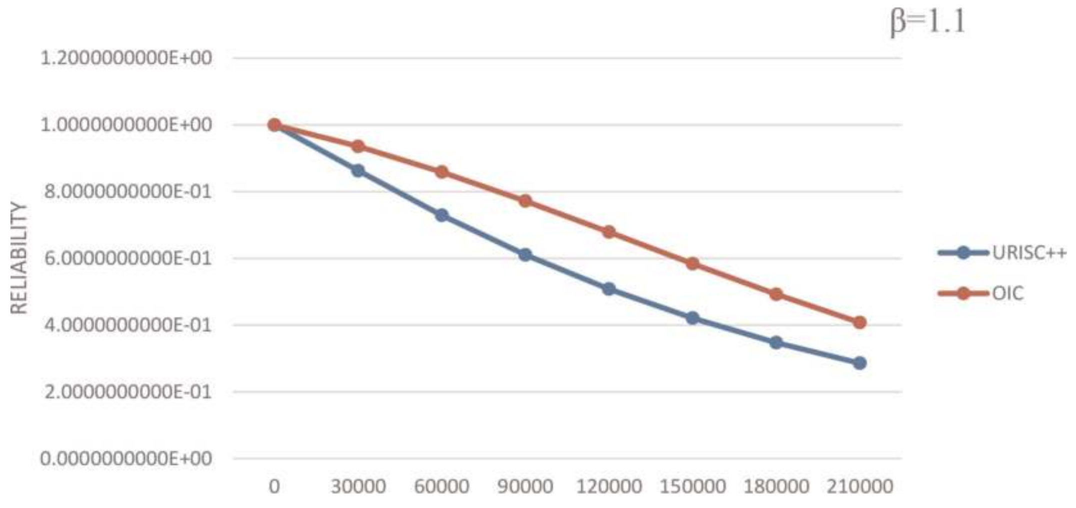


Figure 16.
β = 1.1 reliability vs. time (hours).

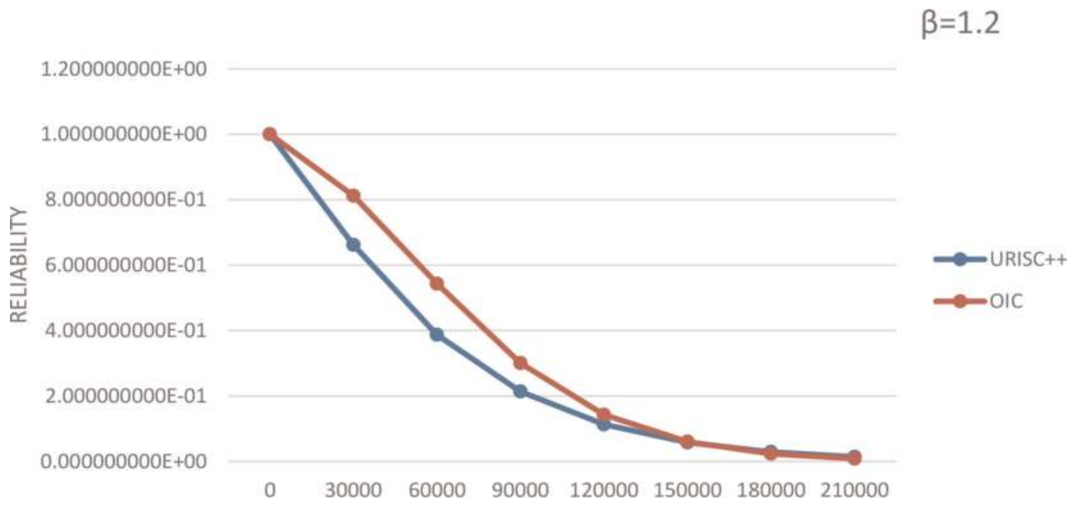


Figure 17.
β = 1.2 Reliability vs. time (hours).

reliability gap widens between 32-bit OIC and URISC++ when β = 1.1 (**Figure 16**) after 60,000 hours or 6.84 years. The reliability levels of OIC fall below that of URISC++ because single component reliability reduces below 0.5 after 23.4 years as shown in **Figure 17** and the redundancy in the OIC does not have any merit thereafter.

8. Conclusion

1. Power, area and total power for OIC and for its contender URISC++ are evaluated. OIC consumes less power and area compared to its contender. The registers count in OICs is significantly less compared to URISC++. It is observed that two large register files in URISC++ consume more power, unlike OIC which does not maintain register files.

2. The performance overheads at instruction level and application level are evaluated. In terms of performance overhead, based on the analysis in the Section 5, performance loss is incurred in compute intensive and memory intensive micro-benchmarks mainly due to MUL and DIV instructions in the programs. But the performance loss will not be high in programs with right mix of arithmetic instructions.

3. In 1:1 configuration of multi-core system with OICs i.e., one conventional core with one OIC, all the emulation request from the conventional core is handled by OIC. In 2:1 configuration (two cores and one OIC), simultaneous failures in two conventional cores results in higher performance loss for the application executing in the system. This performance loss can be reduced by augmenting the multi-core configuration with an additional OIC. That is, 1:1 model proves to be a viable solution with minimal performance loss. This is validated by the simulation results presented in this chapter. On 1:1 and 1: N basis i.e., one MIPS core with one or more OICs can scale to 100 MIPS core with 100 or more OICs. Hence, MCS-OIC model is a scalable design alternative.

4. As expected, it is observed from the reliability analysis of OIC that an increase in the number of subtractors results in higher reliability. Alternatively, it can be understood that replication of functional units improves reliability of the OIC significantly. Hence, TMR + SCS mode has higher reliability compared to the other modes.

5. The yield of the fault tolerant die is slightly lesser than the original die for all the design alternatives of MCS-OIC. It is inferred that larger chips with increasing redundancy widens gap between the yield of the original dies and fault tolerant dies. Thus, a trade-off exists between the die yield and fault tolerance provided by the design alternatives (discussed above) having redundancy ranging between 2% and 11%.

6. Reliability of OIC and URISC++ are evaluated and compared. Evaluation results indicate that OIC is more reliable than URISC++ both in the defect induced phase and the wear out induced phase. It can be understood that the level of redundancy is significantly less in URISC++ compared to OIC.

Author details

Shashikiran Venkatesha[1]* and Ranjani Parthasarathi[2]

1 Vellore Institute of Technology, Vellore, Tamil Nadu, India

2 Department of Information Science and Technology, College of Engineering Guindy, Anna University, Chennai, Tamil Nadu, India

*Address all correspondence to: shashikiran.annauniv@gmail.com

References

[1] Borkar S. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. IEEE Micro. 2005;**25**(6): 10-16

[2] Shivakumar P, Kistler M, Keckler SW, Burger D, Alvisi L. Modeling the effect of technology trends on the soft error rate of combinational logic. In: Proceedings of International Conference on Dependable Systems and Networks. IEEE Explorer. 2002. pp. 389-398. DOI: 10.1109/DSN.2002.1028924

[3] Feng S, Gupta S, Ansari A, Mahlke S. Shoestring: Probabilistic soft error reliability on the cheap'. ACM SIGARCH Computer Architecture News. 2010; **38**(1):385-396

[4] Li T, Ambrose JA, Ragel R, Parameswaran S. Processor design for soft errors: Challenges and state of the art. ACM Computing Surveys. 2016; **49**(3):1-44

[5] Mittal S. A survey of techniques for managing and leveraging caches in GPUs. Journal of Circuits, Systems, and Computers. 2014;**23**(08):1430002

[6] Rusu S, Muljono H, Ayers D, Tam S, Chen W, Martin A, et al. 5.4 Ivytown: A 22 nm 15-core enterprise Xeon® processor family. In: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). IEEE Explorer; 2014. pp. 102-103. DOI: 10.1109/ISSCC.2014.6757356

[7] Zyuban V, Taylor SA, Christensen B, Hall AR, Gonzalez CJ, Friedrich J, et al. IBM POWER7+ design for higher frequency at fixed power. IBM Journal of Research and Development. 2013;**57**(6): 1-1

[8] Postman J, Chiang P. A survey addressing on-chip interconnect: Energy and reliability considerations. International Scholarly Research Notices. 2012;**2012**:1-9. Article ID: 916259. DOI: 10.5402/2012/916259

[9] Nassif SR, Mehta N, Cao Y. A resilience roadmap. In: 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010). IEEE Explorer; 2010. pp. 1011-1016. DOI: 10.1109/DATE.2010.5456958

[10] Karnik T, Tschanz J, Borkar N, Howard J, Vangal S, De V, et al. Resiliency for many-core system on a chip. In: 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE Explorer; 2014. pp. 388-389. DOI: 10.1109/ASPDAC.2014.6742921

[11] Gaisler J. A portable and fault-tolerant microprocessor based on the SPARC v8 architecture. In: Proceedings International Conference on Dependable Systems and Networks. IEEE Explorer; 2002. pp. 409-415. DOI: 10.1109/DSN.2002.1028926

[12] Lin S, Kim YB, Lombardi F. Design and performance evaluation of radiation hardened latches for nanoscale CMOS. IEEE Transactions on Very Large-scale Integration Systems. 2010;**19**(7): 1315-1319

[13] Slayman CW. Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations. IEEE Transactions on Device and Materials Reliability. 2005;**5**(3):397-404

[14] Pomeranz I, Vijaykumar TN. FaultHound: Value-locality-based soft-fault tolerance. In: Proceedings of the 42nd Annual International Symposium

on Computer Architecture. ACM Digital Library; 2015. pp. 668-681. DOI: 10.1145/2749469.2750372

[15] Meaney PJ, Swaney SB, Sanda PN, Spainhower L. IBM z990 soft error detection and recovery. IEEE Transactions on Device and Materials Reliability. 2005;**5**(3):419-427

[16] Stackhouse B, Bhimji S, Bostak C, Bradley D, Cherkauer B, Desai J, et al. A 65 nm 2-billion transistor quad-core Itanium processor. IEEE Journal of Solid-State Circuits. 2008;**44**(1):18-31

[17] Venkatesha S, Parthasarathi R. 32-Bit one instruction core: A low-cost, reliable, and fault-tolerant core for multicore systems. Journal of Testing and Evaluation. 2019;**47**(6):3941-3962. DOI: 10.1520/JTE20180492. ISSN 0090-3973

[18] Hamming RW. Error detecting and error correcting codes'. The Bell System Technical Journal. 1950;**29**(2):147-160

[19] Rajendiran A, Ananthanarayanan S, Patel HD, Tripunitara MV, Garg S. Reliable computing with ultra-reduced instruction set co-processors. In: DAC Design Automation Conference 2012. ACM Digital Library; 2012. pp. 697-702. DOI: 10.1145/2228360.2228485

[20] Ananthanarayan S, Garg S, Patel HD. Low-cost permanent fault detection using ultra-reduced instruction set co-processors. In: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE Explorer; 2013. pp. 933-938. DOI: 10.7873/DATE.2013.196

[21] Sundaramoorthy K, Purser Z, Rotenberg E. Slipstream processors: Improving both performance and fault tolerance'. ACM SIGPLAN Notices. 2000;**35**(11):257-268

[22] LaFrieda C, Ipek E, Martinez JF, Manohar R. Utilizing dynamically coupled cores to form a resilient chip multiprocessor. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). IEEE Explorer; 2007. pp. 317-326. DOI: 10.1109/DSN.2007.100

[23] Aggarwal N, Ranganathan P, Jouppi NP, Smith JE. Configurable isolation: Building high availability systems with commodity multi-core processors. ACM SIGARCH Computer Architecture News. 2007;**35**(2):470-481

[24] Smolens JC, Gold BT, Falsafi B, Hoe JC. Reunion: Complexity-effective multicore redundancy. In: 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06). IEEE Explorer; 2006. pp. 223-234. DOI: 10.1109/MICRO.2006.42