Impact of failures in a MPSoC with shared coprocessors to extend the RISC-V ISA

Jorge Reis
jorge.reis@lesc.ufc.br
PPGETI-UFC Federal University of Ceará
Fortaleza, Ceará, Brazil

Jarbas Silveira
jarbas@lesc.ufc.br
PPGETI-UFC Federal University of Ceará
Fortaleza, Ceará, Brazil

César Marcon
cesar.marcon@pucrs.br
PPGCC-PUCRS Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Rio Grande do Sul
Brazil

ABSTRACT

Reduced Instruction Set (RISC) architectures optimize a complex ISA by implementing only the most frequently used instructions in hardware; however, the application execution time significantly increases when executing heavily used instructions in software. One technique that optimizes the trade-off of implementation cost and execution time is the use of a Multiprocessor System-on-Chip (MPSoC), in which RISC processors extend their ISA by sharing coprocessors that implement lesser-used instructions. This article analyses the impact of shared coprocessor failures on two RISC-V MPSoC architectures. We evaluated these architectures using two image processing applications and four failure rates in terms of power dissipation, energy consumption, area consumption, maximum operating frequency, and execution time. The experiments show a 16% maximum increase in execution time for the application with a low percentage of instructions executed. In contrast, for the application with the highest rate of coprocessor use, considering a one-fault scenario, the execution time does not increase significantly in one of the architectural configurations proposed for the MPSoC.

CCS CONCEPTS

• **Computer systems organization → Fault-tolerant network topologies**; *Redundancy*; *System on a chip.*

KEYWORDS

RISC-V, ISA, MPSoC shared resources, NoC, Fault Tolerance

ACM Reference Format:
Jorge Reis, Jarbas Silveira, and César Marcon. 2022. Impact of failures in a MPSoC with shared coprocessors to extend the RISC-V ISA. In *11th Latin-American Symposium on Dependable Computing (LADC 2022), November 21–24, 2022, Fortaleza/CE, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3569902.3569906

LADC 2022, November 21–24, 2022, Fortaleza/CE, Brazil

ACM ISBN 978-1-4503-9737-7/22/11…$15.00
https://doi.org/10.1145/3569902.3569906

1 INTRODUCTION

Underutilized instructions in an Instruction Set Architecture (ISA) can increase area and power costs if implemented in hardware or increase execution time if implemented by software routines [6][1][3]. The Reduced Instruction Set Computer (RISC) architecture is a solution for reducing the ISA and simplifying hardware implementation.

RISC-V is a modular ISA that includes a 32-bit standard ISA called RV32I that can perform all computer functions with only integer and control instructions and emulate other modular extensions by software routines [8]. But while this reduces the complexity of the CPU, the program execution time increases. So, to boost RV32I processing power, these extensions can also be implemented in hardware. This modular design of RISC-V provides different approaches to implementing an ISA. Nevertheless, a RISC-V processor can have an underutilized extension implemented on hardware.

Several studies [5] [2] [7] [4] show that a way to minimize these problems is by utilizing a Multiprocessor System-on-Chip (MPSoC) to extend the ISA among multiple processors. This work proposes an MPSoC using arithmetic coprocessors to extend the ISA of multiple low-complexity 32-bit RISC-V processors (RV32I), sharing them to implement the modular extension RV32M and what happens to them when some of these coprocessors fail.

The remaining of this article is organized as follows. Section II gives an overview of related works. In Section III, we present the target architecture of this work describing the elements of the MPSoC and the fault models. Subsequently, Section IV explains the metrics and benchmarks in our experimentation. Finally, Section V discusses the results, and Section VI summarizes the conclusions of this work.

2 RELATED WORK

This paper proposes to expand the RISC-V ISA using RV32I processors and shared coprocessors implementing the extension connected through an NoC, as presented in Lima et al. [5], and analyze the outcomes in the system when part of the coprocessors starts to fail.

Lima et al. [5] address this issue by presenting an MPSoC that allows ISA to extend multiple RV32I processors, simultaneously sharing system resources. This way, the hardware cost decreases at a low penalty for the program execution time. Four different scenarios were analyzed in their work, comparing area and energy consumption, power dissipation, and execution time. Unlike our work, they do not examine the impact of failures on the coprocessors shared amongst the RV32I processors.

Becker et al. [2] propose to trim off the support for costly ISA extensions in some cores of the MPSoC, transforming them into Partial-ISA cores. The proposal was tested using the Floating Point operations of a RISC-V ISA extension as a case study, evaluating power and area reduction with real-life processor descriptions. Their results showed that it is possible to reduce energy consumption with a better energy-delay trade-off reducing the support for underutilized instructions, which is done in our work by sharing the coprocessors that implement them as opposed to trimming them.

Vieira et al. [7] propose a Single-ISA asymmetric multicore architecture to combine high performance and energy efficiency in the same chip by providing different microarchitectures so the applications can transparently migrate from one to another accordingly.

This work proposes a mechanism that optimizes the use of hardware resources to have an energy-efficient MPSoC, supporting a complex ISA set through coprocessors. Unlike [2], this work has a similar approach to the one our work proposes: optimizing the implementation of the underutilized instructions.

3 TARGET ARCHITECTURE

This work expands the RV32I architecture to RV32IM by implementing the RV32M extension instructions in the multiplication and division coprocessors shared between the MPSoC RV32I processors. Among the various configurations of coprocessor organization analyzed by Lima et al. [5], FineGrain Coprocessing (FGC) obtained the best cost-benefit among area consumption, power dissipation, and execution time; therefore, we chose this configuration to analyze the fault-tolerant architecture.

The target architecture is a 3D MPSoC in a 4x4x2 configuration with 16 RV32I processors, eight multiplication, and eight division coprocessors. All coprocessors are shared by MPSoC processors that execute RV32M extension instructions, sending packets with the instruction and data and receiving back packets containing the result.

In a no-fault scenario, each coprocessor is shared by only two processors. However, in fault scenarios, the workload of each failed coprocessor executes in a no-fault coprocessor.

3.1 Processor

The target architecture employs a single-cycle processor implementation of the RISC-V architecture that supports the RV32I and RV32M ("M" extension) ISAs. The RV32I configuration can perform multiplication and division operations via software to emulate the RV32M. The RV32M extension adds four multiplication operations (MUL, MULH, MULHSU, and MULHU) and four division operations (DIV, DIVU, REM, and REMU) to the RV32I ISA.

The RV32M extension in coprocessors external to the datapaths of processors that implement the RV32I architecture allows programs to be compiled for an RV32IM architecture.

3.2 NoC

We use a 3D NoC with 4x4x2 regular mesh topology, XYZ routing, wormhole packet switching, on-off flow control, and buffering at the input ports. It also features fixed priority and round-robin scheduling. A router with six ports (north, south, east, west, up, and down), a Network Interface (NI), and a PE with its local memory make up each NoC tile. The NI is in charge of interfacing the router and PE by packing and unpacking data. Figure 1 displays the structure of an NoC tile. Each PE can be a processor, as shown in Figure 1a, a divider arithmetic coprocessor, as shown in Figure 1b, or a multiplier arithmetic coprocessor, as shown in Figure 1c.

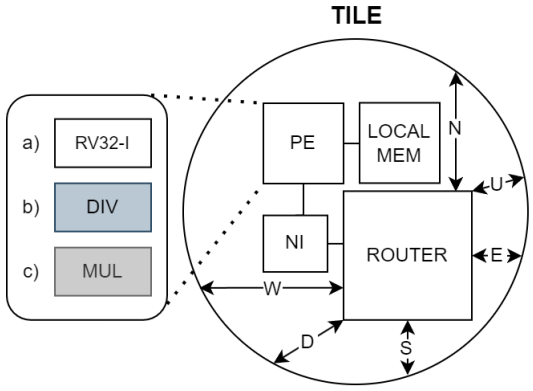

Figure 1: Configuration of the MPSoC tiles.

3.3 Architectural Organization

Figure 2 shows that the MPSoC processing elements are organized into FGC and FGC_MIX configurations.

In the FGC configuration, the first layer has only RV32I processors, and the second has multiplication and division coprocessors. In this configuration, the processors share the two closest multiplication and division coprocessors, keeping the maximum load distribution at two processors per coprocessor. For example, in Figure 2a, processors 0 and 1 share coprocessors 16 and 17, while processors 6 and 7 share coprocessors 22 and 23.

In the FGC_MIX configuration, the first layer has RV32I processors and division coprocessors, while the second has RV32I processors and multiplication coprocessors, as illustrated in Figure 2b. In this way, each processor has a multiplication and division coprocessor one hop away, reducing the time spent by traffic in the NoC. For example, processors 1 and 15 share coprocessors 0 and 17, and it is possible to notice that the two are one hop away from their coprocessors in this configuration.

3.4 Fault Models

To analyze the impact of shared coprocessor failures in the system, we choose two, six, and eight coprocessors (12.5%, 37.5% and 50% of the total coprocessors), injecting faults before runtime. For each fault rate, half is from the multiplication arithmetic coprocessor and the other half from the division arithmetic coprocessor. Moreover, we selected a spare coprocessor to handle the workload of each failed coprocessor to ensure that processes do not fail due to these crashes.

For each number of faults, we randomly chose coprocessors in three models designated as A, B, and C. There are nine-fault scenarios for FGC, nine-fault scenarios for FGC_MIX, and one scenario without faults for each configuration, totaling 20 scenarios

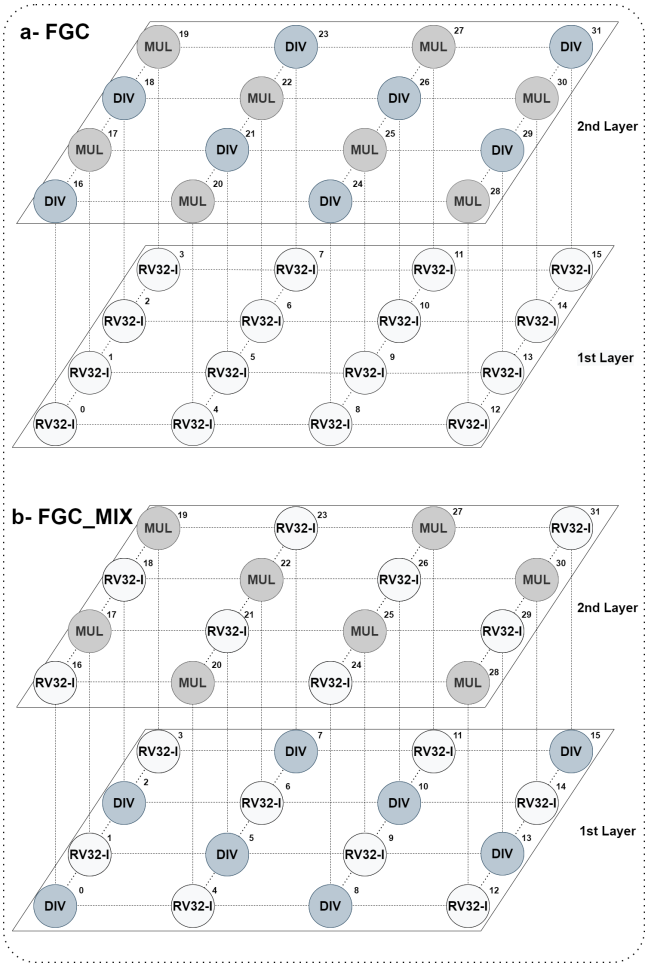

Figure 2: Proposed architectural organization of the MPSoC.

regarding FGC or FGC_MIX configurations, number of faults (0F, 2F, 6F, or 8F), and model (A, B or C).

After the fault selection process for the simulation, its replacement is selected according to the following criteria:

- Must be the same type as the failed coprocessor;
- Must be chosen from the options with the fewest hops away for the processors affected by the failure;
- Not yet being used as a replacement for another failed coprocessor.

According to these criteria, each coprocessor chosen as a replacement is shared by a maximum of four processors. It is important to emphasize that this work aims to evaluate the behavior of the MPSoC RV32I processors by changing the number of shared coprocessors to run the RV32M extension and not structural NoC faults. Therefore, the NoC topology is kept intact; only the coprocessor address used in each affected processor is changed, enabling the processors to send instructions to the replacement coprocessors.

In FGC_MIX configuration fault scenarios, the best replacement case has the substitute coprocessor one hop away from an affected processor and three hops away from the other. While in the FGC configuration, the best replacement case has the substitute coprocessor two hops away from an affected processor and three hops away from the other.

Figure 3 illustrates the 6F FGC_MIX B scenario displaying the six fault coprocessors and the chosen substitutes. The multiplication coprocessor 22 used by RV32I numbers 6 and 23 in the first and second layers failed. The chosen replacement pointed by the red arrow is the multiplier 19 in the second layer because it followed the determined criteria of being one hop away from the 23 processors and three hops away from the 6. One case in which this does not occur is for processors 12 and 29, in which the division 13 coprocessor shared by them fails, and the only one that meets the established criteria is number 15, which is three hops from each other. For processor number 12, the situation is more critical because the multiplier shared by both fails, and the substitute is three hops away. Therefore, this processor takes the most significant number of MPSoC cycles to complete its part of the application.

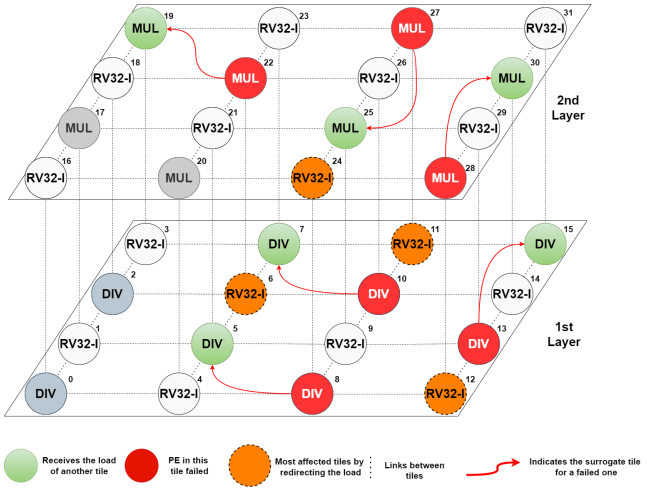

Figure 3: Fault scenario 6F FGC_MIX B.

4 METRICS AND EXPERIMENTS

4.1 Metrics

The metrics used to evaluate the experiments are the following:

- The total area of the synthesized MPSoC, obtained by summing cells and connections between these cells;
- Dissipated power;
- Maximum operating frequency reached in the synthesis;
- Number of cycles required for the program execution;
- Energy consumed in each scenario regarding a benchmark.

4.2 Benchmarks

All scenarios were analyzed using Contrast and Conv digital image processing programs. Both programs use the sixteen processors available in the MPSoC, dividing the task of processing each part of the image among the processors.

The Contrast benchmark is a program that receives an image as input and adjusts its contrast. The multiplication and division

instructions are 2.1% and 1.4% of the total number of instructions, respectively. The Conv benchmark uses the Sobel filter to convolve a mask against an input image. This benchmark has 5.6% of multiplication instructions from the total number of instructions, and there are no division operations.

Figure 4 exemplifies the flowchart of the applications. A python script generates a header containing the datasets to be processed and one to be used as a reference. The application divides the image processing among the MPSoC threads, and after finishing the processing, it compares the obtained result with the reference dataset, returning success or failure.

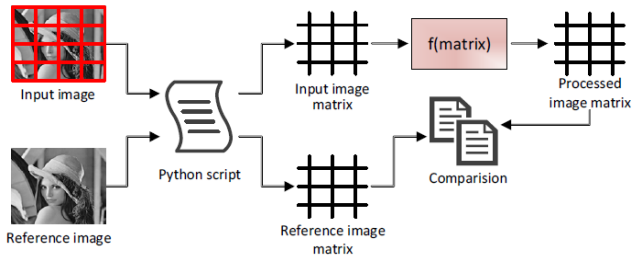

Figure 4: Flowchart of developed applications.

4.3 Evaluation Procedure

Scenarios were simulated and validated with Verilator; the results related to the synthesis metrics were obtained with the Genus Synthesis Solution, configured for a 65 nm CMOS technology and 100 MHz operating frequency. Reports on power dissipation, execution time, operating frequency, and area consumption are presented in the next section.

5 EXPERIMENTAL RESULTS

Table 1 displays the synthesis results of power dissipation, area consumption, and maximum operating frequency for both architectural MPSoC configurations. The synthesis results are similar, as the difference between the two scenarios is only the placement of tiles.

Table 1: Synthesis Results

Scenario	Power (mW)			Area	Freq
	Leakage	Dynamic	Total	(mm2)	(MHz)
FGC	32.508	203.065	235.574	4.13	100.0
FGC_MIX	32.398	204.635	237.033	4.12	100.0

Table 2 displays the data related to the execution time of the benchmarks. N_Cycles is the number of cycles required to complete the benchmark run, INC(%) is the percentage increase in the number of cycles compared to the no-fault scenarios, and std_dev is the standard deviation of the FGC and FG_MIX scenarios by the number of failures. The execution time in ms is calculated using the number of cycles obtained by the simulation and the operating frequency calculated with the synthesis results.

In the 0F FGC_MIX scenario, the execution of the Contrast application is completed with several cycles, only 3.7% less than in the 0F FGC scenario. In the Conv application, the 0F FGC_MIX scenario is executed with several cycles 15% less than the FGC configuration.

In the three 2F FGC scenarios, in the Contrast application, the increase in the number of cycles presented more uniform values than the 2F FGC_MIX scenarios, which have a higher standard deviation. However, all FGC_MIX scenarios continue with a shorter execution time than the FGC scenarios.

The Conv application shows that the percentage increase in the number of cycles is significantly higher than in the Contrast application, as the percentage of instructions used in the RV32M extension is higher in the Conv application. In this way, the load of packets sent by the CPU to the coprocessors is more significant, which increases the weight of the number of cycles spent on packet traffic on the network.

Furthermore, the behavior observed in the 0F scenarios is the opposite of that observed in the 2F scenarios, with all FGC configurations having a shorter execution time than the FGC_MIX configurations. This opposite behavior happens because the FGC_MIX configuration has a lower number of hops among RV32I tiles, and the number of MUL or DIV tiles is minimized with the increase in the number of hops between the processors affected by failure and replacement coprocessors. In the three 2F FGC scenarios, the increase in the number of cycles presented more uniform values than the FGC_MIX scenarios with a higher standard deviation.

With six coprocessors failing, both scenarios had a high standard deviation value in the Contrast application. The FGC A scenario presented the same execution time as the FGC_MIX B scenario, while the other FGC scenarios continued with a shorter execution time than the FGC_MIX scenarios.

In the Conv application, the standard deviation of the FGC scenarios was high mainly because of the 6F FGC B scenario, which presented a much lower execution time than the others, with a value close to that of the 2F FGC B scenario. This happened because all route deviations for replacement coprocessors could get the lowest number of hops for replacement, unlike the 6F FGC A E C scenarios.

The data uniformity of the two applications of the 8F scenarios occurs because, in all scenarios, some processors did not obtain the lowest possible number of hops for their replacement coprocessor. The behavior observed in the FGC_MIX scenarios of having a shorter execution time, even if not so significant, is inherent in the configuration having a smaller number of hops of the distance between the source-destination pairs.

We notice some 2F scenarios with execution times of the same magnitude as those with 6F or 8F. For example, we see similar execution times in 2F FG_MIX C and 6F FGC_MIX C scenarios and 2F FGC_MIX A and 8F FGC_MIX B, regarding Contrast and Conv applications, respectively. Figures 5 and 6 clarify these cases showing the number of cycles required for each MPSoC processor to complete the benchmark execution, allowing a more detailed analysis of the 2F FGC_MIX A and 8F FGC_MIX B scenarios.

Table 2 displays that the 2F scenario completes the Conv benchmark with some cycles of the same magnitude as the 8F scenario, even though the number of failures has quadrupled. This execution

Table 2: Execution Time

SCENARIO		CONTRAST				CONV			
		N CYCLES	INC(%)	Execution time (ms)	std_dev	N CYCLES	INC (%)	Execution time (ms)	std_dev
0F	FGC	8438384		84.4	0	3219214		32.2	0
	FGC_MIX	8133248		81.3	0	2731342		27.3	0
2F	FGC A	8944496	6%	89.4		3726437	16%	37.3	
	FGC B	8928072	6%	89.3	0.54	3722953	16%	37.2	0.02
	FGC C	9050182	7%	90.5		3721989	16%	37.2	
	FGC_MIX A	8851166	9%	88.5		4035044	48%	40.4	
	FGC_MIX B	8782510	8%	87.8	1.02	4039401	48%	40.4	0.13
	FGC_MIX C	9023972	11%	90.2		4010862	47%	40.1	
6F	FGC A	9344860	11%	93.4		4162851	29%	41.6	
	FGC B	9251569	10%	92.5	2.05	3736972	16%	37.4	2.18
	FGC C	9725494	15%	97.2		4227448	31%	42.3	
	FGC_MIX A	8820825	8%	88.2		4047167	48%	40.5	
	FGC_MIX B	9344838	15%	93.4	2.37	4049401	48%	40.5	0.02
	FGC_MIX C	8864678	9%	88.6		4052806	48%	40.5	
8F	FGC A	9733104	15%	97.3		4152418	29%	41.5	
	FGC B	9744527	15%	97.4	0.08	4162874	29%	41.6	0.05
	FGC C	9725494	15%	97.2		4162253	29%	41.6	
	FGC_MIX A	9411959	16%	94.1		4026550	47%	40.3	
	FGC_MIX B	9434138	16%	94.3	0.12	4024374	47%	40.2	0.22
	FGC_MIX C	9406855	16%	94.1		4072358	49%	40.7	

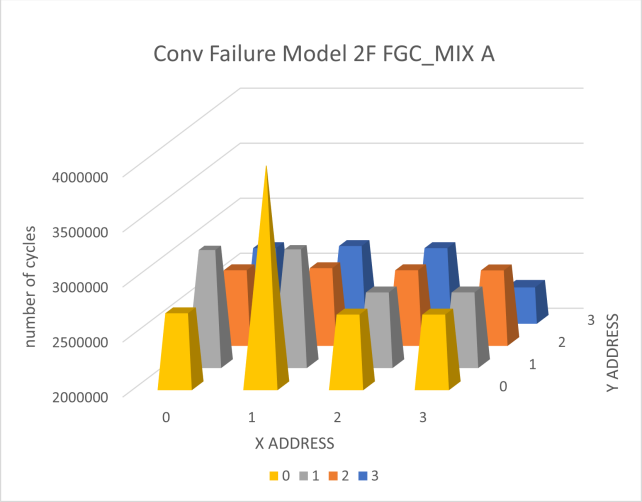

Figure 5: Number of cycles to complete benchmark execution for each processor.

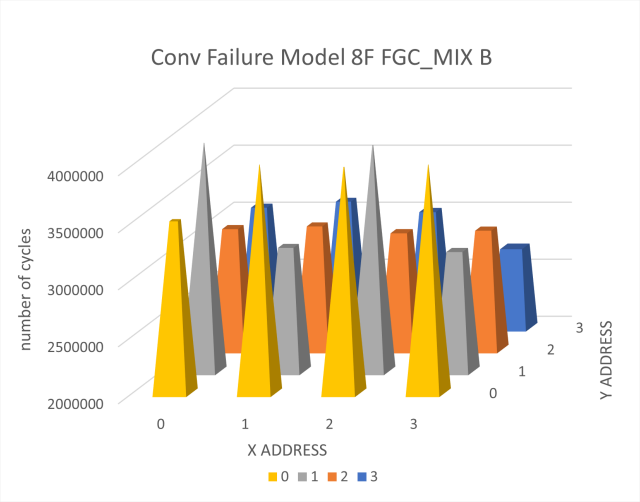

Figure 6: Number of cycles to complete benchmark execution for each processor.

time happens because the number of cycles increases across the MPSoC processors as the number of failures increases.

Table 3 displays the energy consumption of running the Contrast and Conv applications in mJ. The table also shows the percentage increase in energy consumption for each scenario compared to the equivalent 0F scenario. This increase is proportional to the rise in the number of cycles.

Both 0F FGC and FGC_MIX scenarios present similar energy consumption in the Contrast application, whereas, for 2F scenarios, the increase was 6-11% compared to equivalent 0F scenarios. For the 6F scenarios, the increase ranges from 8.5% to 15.3%. Finally, for scenarios with 8F, the increase ranges from 15.3% to 16%. Important to note that all scenarios reach a saturation level at 8F, but the 6F FGC C scenario reached this saturation earlier.

Table 3: Energy Consumption of each Scenario

SCENARIO	CONTRAST		CONV	
	Energy (mJ)	Inc (%)	Energy (mJ)	Inc (%)
0F FGC	19.9		7.6	
0F FGC_MIX	19.3		6.5	
2F FGC A	21.1	6.0	8.8	16.0
2F FGC B	21.0	6.0	8.8	16.0
2F FGC C	21.3	7.0	8.8	16.0
2F FGC_MIX A	21.0	9.0	9.6	48.0
2F FGC_MIX B	20.8	8.0	9.6	48.0
2F FGC_MIX C	21.4	11.0	9.5	47.0
6F FGC A	22.0	11.0	9.8	29.0
6F FGC B	21.8	10.0	8.8	16.0
6F FGC C	22.9	15.0	10.0	31.0
6F FGC_MIX A	20.9	8.0	9.6	48.0
6F FGC_MIX B	22.2	15.0	9.6	48.0
6F FGC_MIX C	21.0	9.0	9.6	48.0
8F FGC A	22.9	15.0	9.8	29.0
8F FGC B	23.0	15.0	9.8	29.0
8F FGC C	22.9	15.0	9.8	29.0
8F FGC_MIX A	22.3	16.0	9.5	47.0
8F FGC_MIX B	22.4	16.0	9.5	47.0
8F FGC_MIX C	22.3	16.0	9.7	49.0

For the Conv application in 0F scenarios, the FGC topology consumes approximately 14% more power than FGC_MIX. For the 2F scenarios, the FGC_MIX settings consumed 7% more energy than the FGC_MIX settings, inverting the 0F scenario. For the 6F scenarios, the FGC A and C scenarios increase the energy consumption in percentage compared to the 2F scenario, unlike the FGC_MIX and FGC B scenarios, which did not have a significant increase in energy consumption. Finally, in the 8F scenario and the Contrast application, the energy consumption reaches a saturation level; thus, the FGC_MIX and FGC scenarios had similar energy consumption.

6 CONCLUSIONS

This paper analyzed the impact of failures on the coprocessors of a system that extends the ISA from multiple RISC-V processors using shared coprocessors in an MPSoC. We created 20 test scenarios with four faulty rates (0, 12.5%, 37.5%, and 50%), two architectures for the MPSoC (FGC and FGC_MIX), and three different fault scenarios (a, b and c).

The Contrast application has a lower percentage of instructions on the RV32M extension and therefore depends less from the MPSoC arithmetic coprocessors. So with up to 50% of coprocessors failing, the execution time and energy consumption increase was only 15-16%.

The Conv application, which has a higher percentage of instructions from the RV32M extension and therefore uses the MPSOC arithmetic coprocessors more, with 12.5% of the coprocessors failing, the increase in execution time already reached 16% for the FGC configuration and 48% for FGC_MIX. With 50% of failures in the coprocessors, the number of cycles and energy consumption continue to increase, reaching 29% in the FGC scenarios, and remains without a significant increase in the FGC_MIX scenarios.

Future work includes detecting the failures and analyzing the costs of the replacement strategy during runtime.

REFERENCES

[1] Rafael Auler and Edson Borin. 2017. The Case for Flexible ISAs: Unleashing Hardware and Software. *2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)* (2017), 65–72.

[2] Pedro Henrique Exenberger Becker, Jeckson Dellagostin Souza, and A. C. S. Beck. 2020. Tuning the ISA for increased heterogeneous computation in MPSoCs. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2020), 1722–1727.

[3] Sandeep Dasgupta, Daejun Park, Theodoros Kasampalis, Vikram S. Adve, and Grigore Rosu. 2019. A complete formal semantics of x86-64 user-level instruction set architecture. *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (2019).

[4] Mahmoud A. Elmohr, Ahmed S. Eissa, Moamen Ibrahim, Mostafa Khamis, Sameh El-Ashry, Ahmed Shalaby, Mohamed Abdelsalam, and Mohamed Watheq El-Kharashi. 2018. RVNoC: A Framework for Generating RISC-V NoC-Based MPSoC. *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)* (2018), 617–621.

[5] Pedro Lima, Caio Vieira, Jorge Reis, Alexandre Almeida, Jarbas A. N. Silveira, Roger C. Goerl, and César A. M. Marcon. 2020. Optimizing RISC-V ISA Usage by Sharing Coprocessors on MPSoC. *2020 IEEE Latin-American Test Symposium (LATS)* (2020), 1–5.

[6] Bruno Cardoso Lopes, Rafael Auler, Luiz Ramos, Edson Borin, and Rodolfo Azevedo. 2015. SHRINK: Reducing the ISA complexity via instruction recycling. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. 311–322. https://doi.org/10.1145/2749469.2750391

[7] Caio Vieira and Antonio Carlos Schneider Beck. 2021. Improving energy efficiency by transparently sharing SIMD Execution Units in Assymetric Multicores. *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)* (2021), 1–6.

[8] Andrew WATERMAN and Krste ASANOVIC. 2017. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA , Document Version 2.2.* RISC-V Foundation.