ISA-DTMR

Selective Protection in Configurable Heterogeneous Multicores

Erichsen, Augusto G.; Sartor, Anderson L.; Souza, Jeckson D.; Pereira, Monica M.; Wong, Stephan; Beck, Antonio C.S.

DOI
10.1007/978-3-319-78890-6_19

Publication date
2018

Document Version
Final published version

Published in
Applied Reconfigurable Computing

Citation (APA)
Erichsen, A. G., Sartor, A. L., Souza, J. D., Pereira, M. M., Wong, S., & Beck, A. C. S. (2018). ISA-DTMR: Selective Protection in Configurable Heterogeneous Multicores. In N. Voros, M. Huebner, G. Keramidas, D. Goehringer, C. Antonpoulos, & P. C. Diniz (Eds.), *Applied Reconfigurable Computing: Architectures, Tools, and Applications - 14th International Symposium, ARC 2018, Proceedings* (pp. 231-242). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 10824). Springer. https://doi.org/10.1007/978-3-319-78890-6_19

ISA-DTMR: Selective Protection in Configurable Heterogeneous Multicores

Augusto G. Erichsen[1(✉)], Anderson L. Sartor[1], Jeckson D. Souza[1], Monica M. Pereira[2], Stephan Wong[3], and Antonio C. S. Beck[1]

[1] Institute of Informatics, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
{agerichsen,alsartor,jeckson.souza,caco}@inf.ufrgs.br
[2] Department of Computer Science and Applied Mathematics, Universidade Federal do Rio Grande do Norte (UFRN), Natal, Brazil
monicapereira@dimap.ufrn.br
[3] Computer Engineering Laboratory, Faculty of EEMCS, Delft University of Technology, Delft, The Netherlands
j.s.s.m.wong@tudelft.nl

Abstract. The well-known Triple Modular Redundancy (TMR), when applied to processors to mitigate the occurrence of faults, implies that all applications have the same level of criticality (since they are all equally protected) and are executed in a homogeneous environment, which naturally would waste precious resources in terms of area and energy. However, many current systems are composed of heterogeneous cores that implement the same ISA (e.g., ARM's big.LITTLE or DynamIQ), executing some applications that may be more critical than others and that would require different levels of protection. With that in mind, we propose ISA-DTMR, a non-intrusive approach that, taking advantage of heterogeneous systems, can protect applications at different levels in a totally transparent fashion. By using heterogeneous multicore configurations composed of configurable processors that implement the same Instruction Set Architecture (ISA), we will show that it is possible to adapt the level of protection for each application according to its reliability requirements. When compared to homogeneous processors, ISA-DTMR reduces area by up to 54.9%, and energy consumption by 30.35%, with negligible overhead on performance, for a configuration that balances performance and energy consumption. ISA-DTMR is able to provide the same level of protection for critical applications and even improve the reliability for non-critical applications.

Keywords: Fault tolerance · Heterogeneous architecture · TMR DMR

1 Introduction

The evolution of integrated circuit manufacturing technology has also increased its susceptibility to failure, which can be caused by many factors such as variability, aging, and radiation effects [4]. Many fault tolerance techniques have

N. Voros et al. (Eds.): ARC 2018, LNCS 10824, pp. 231–242, 2018.
https://doi.org/10.1007/978-3-319-78890-6_19

been developed to enable correct execution or detection of errors, and all of them present some sort of overhead when compared to the unprotected version: area, performance, or energy consumption. TMR is one of the most popular, and works by replicating the circuit that needs protection, checking correctness using a majority voter so the faults can be masked. While it presents a high fault tolerance to Single Event Effects (SEE), it results in a huge area and power dissipation overhead. Most importantly, in the case of an environment composed of General Purpose Processor (GPPs) executing multiple processes, it usually assumes that all processors are the same and that all the applications are critical (thus they all should be protected). On the other hand, the advantage of TMR is that it is completely non-intrusive: it can be implemented without any changes in the underlying microarchitecture, so potentially any Commercial Off-The-Shelf (COTS) GPP can be used.

ARM big.LITTLE and, more recently, DynamIQ [1] brought to mainstream market systems that are composed of different microarchitectures but implement the same ISA so that the same binary can transparently execute on different microarchitectures. To save energy, applications that do not need full performance can migrate from one (big) core to another (LITTLE), which indirectly enables the fabrication of integrated circuits with decreased transistor count and die size. Borrowing from this same idea, one could consider that instead of performance, an application would not need to be fully protected: in a very common scenario where many applications are executing, each of them may have its own requirements when it comes to reliability (some being more critical than others). Therefore, if a non-critical application (i.e., can present errors or will not lead to a system failure even if it is not executed until completion) is executed in a completely protected circuit (e.g., TMR), resources (performance, area, and energy) would be wasted in the same way as applications that would not need full performance executing in a homogeneous multicore environment.

Considering the discussion above, this work proposes ISA-DTMR, which is a new design capable of providing protection only to critical applications, taking advantage of current systems with a homogeneous ISA and heterogeneous microarchitectures, lowering the overheads presented by the original TMR. It keeps the software transparency, since no changes in the source code are necessary; and to the hardware, since no changes in the processor are needed. ISA-DTMR uses a similar idea (but for a different end) as the Design Diversity Redundancy (DDR) methodology [3], in which different designs for the same application are available and work together, increasing fault tolerance.

The general architecture of ISA-DTMR is implemented using configurable processors, following the same concept as the ARM big.LITTLE or DynamIQ: it comprises different processors of the same (ISA) that require different amounts of resources to be implemented and have different trade-offs considering performance, energy and fault tolerance. To better illustrate the approach, Fig. 1 depicts a scenario in which four applications are executed in a heterogeneous quad-core processor. In this example, *App #1* is protected with TMR, *App #2*

and *App #4* with Dual Modular Redundancy (DMR), and *App #3* is executed without any fault tolerance technique.

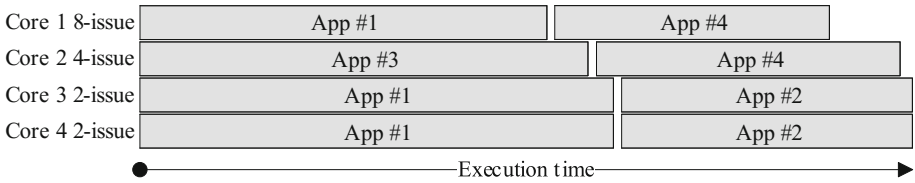


Fig. 1. Applications running in heterogeneous cores

We developed the proposed technique in the VEX ISA, using different designs of the ρ-VEX configurable VLIW processor [27]. The ρ-VEX can be easily configured to have a different number of issue slots (e.g., 2, 4, or 8). Each configuration brings changes to the optimization axes. The 8-issue usually has the best performance for the benchmarks, but greater area occupation and power consumption. The 2-issue, on the other hand, has lower performance in exchange of its reduced power consumption and area occupation. Finally, the 4-issue has better balance between both. We take advantage of these features to build heterogeneous multicore processors, which can have different optimization axes.

The design was synthesized for both ASIC and FPGA. We have compared nine different processor configurations (three homogeneous and six heterogeneous), with two benchmark sets. By applying the proposed approach to a heterogeneous quad-core in the most energy-oriented configuration, it is possible to reduce the area and power dissipation by 54.9% and 57.7%, respectively, when compared to its homogeneous counterpart comprised of 8-issue processors. Considering a particular set of applications, we show that energy is reduced by up to 35.06% with 12.08% of performance overhead. By using a configuration that weighs performance and energy, it is possible to reduce the energy consumption by 30.35% while maintaining almost the same performance. Moreover, all ISA-DTMR designs maintain the same level of protection to critical applications as the conventional TMR when applied to homogeneous processors. Therefore, ISA-DTMR can be used to execute applications with different priorities in terms of protection and, at the same time, save area and energy consumption.

This article is organized as follows. We discuss the state-of-art in Sect. 2. In Sect. 3, the proposed approach is presented and its implementation details are discussed. In Sect. 4, the results are presented in terms of area, power dissipation, energy consumption, performance, and error/failure rates. Section 5 concludes this paper and presents future directions.

2 Related Work

Several works have been proposed for the detection and correction of transient faults in multicore processors. These works aim to improve the fault tolerance

of the target system, typically based on redundancy, which may be implemented in software, hardware, or both. Next, some of these techniques will be discussed.

Software-based techniques that use the compiler to replicate instructions are common both for VLIW processors [5,12,16,24] and for superscalar processors [13]. Unlike compiler instruction replication techniques, ISA-DTMR introduces no extra instructions to the application code, keeping its original size and memory footprint; and does not impose any changes to the original binary (so no extra tools or recompilation are necessary).

Nonetheless, software solutions can also be applied at thread-/process-level. In [14], the authors exploit the parallel capacities of SMT processors to duplicate threads and detect faults via DMR, while in [11] this technique was improved and tested in a dual core, SMT enabled processor, using a higher workload (multiple applications). The authors of [23] take advantage of spare or idle cores in multicore systems to triplicate application processes and apply TMR to detect and correct faults. On the other hand, [6] propose a generic approach for many-core systems, in which a dedicated hardened core controller is responsible for replicating and synchronizing threads for fault detection and recovery by either Duplication With Comparison (DWC) or TMR techniques. Although most of these software-based techniques rely on replicating threads similarly as ISA-DTMR, they do not consider a heterogeneous environment with different levels of criticality for the applications (including the possibility of no protection at all for some of them).

Among the many hardware-based solutions, the authors of [8,9] propose a framework that creates sets of single-ISA cores with different strategies for fault tolerance (fully TMR cores, pipeline, register file and cache TMR). The sets are generated to maximize reliability for different applications, while keeping a power budget, considering future restrictions brought by dark silicon. Then, during runtime, depending on the running application's requirement, a core that gives the necessary protection level is selected, while all the others are kept dark. The authors in [18] trade-off the axes of fault tolerance, energy consumption, and performance by using techniques to duplicate and re-execute faulty instructions, reduce energy consumption by using power gating and control the Instruction-level Parallelism (ILP) of the application so more duplication or energy savings can be achieved. The following works look into these techniques in different granularities [19–21]. In contrast to all those hardware-based solutions, ISA-DTMR requires no modifications inside the processor (i.e., it is not intrusive), so it can be totally based on COTS GPPs. Comparators and voters can be included in many non-intrusive ways, such as in a separate ASIC, in a MPSoC or even by software. Therefore, ISA-DTMR can use existing cores of heterogeneous GPPs to increase reliability, considering individual application criticality to minimize resource usage. Moreover, if the cores are not used for redundancy, they can be used for regular application execution or turned off. This contrasts to typical DMR/TMR solutions, in which all the redundant hardware is permanently used for reliability; or, in the specific case of [9], to the impossibility of using the extra transistors added for fault tolerance to accelerate execution.

Still related to hardware, to reduce the probability of Common Mode Failures (CMF) occurrence, one solution is to provide TMR using different technologies or architectures. This technique is called Diversity TMR (DTMR), or TMR based on DDR [10]. The authors in [25] propose DTMR to investigate protection against CMF in an 8×8 matrix implemented in FPGA. [2] uses DTMR to improve fault tolerance in FPGAs. [26] presents a reliability analysis using design diversity as metric using RISC, VLIW and CGRA architectures to generate the results. Differently from other works in DTMR, ISA-DTMR exploits the binary compatibility between cores in heterogeneous single-ISA multicores. Having the same ISA between cores eases the software development process, since no special compilers or toolchains are necessary to deploy the system.

Therefore, by considering that applications may have different levels of criticality, ISA-DTMR uses the same idea of design diversity as DTMR, but taking advantage of configurable processors to build heterogeneous multicore environments that implement a single ISA. So, in cases some applications do not demand high reliability, it is possible to use the extra cores for parallel execution of other applications or even turn them off to save energy. In this work we reassemble a big.LITTLE-based environment by using different designs of the ρ-VEX family, considering a great number of scenarios. We show how such environment can be employed for fault tolerance and, at same time, balance energy and performance, when criticality is not necessary for all applications.

3 ISA-DTMR Implementation

As already discussed, ISA-DTMR considers the fact that different applications have different needs when it comes to reliability: some are critical and need to be protected against soft errors, while non-critical ones may tolerate erroneous results. ISA-DTMR exploits this difference of criticality by protecting some applications with TMR, others with DMR, and by executing non-critical applications without any fault tolerance mechanism, all concurrently, thus reducing energy and area. Application criticality can be decided at design time, or through dynamic strategies, such as shown in [6].

In this work, the configurable ρ-VEX softcore VLIW processor [27] is used as case study, which is implemented in VHDL and we have full access to its low level description. The ρ-VEX core has a five-stage pipeline, and it can be easily configured to have a different number of issue slots (e.g., 2, 4, or 8). Each pipelane (issue-slot) may contain different functional units from the following set: ALU (always present), multiplier, memory, and branch units. The ISA-DTMR implementation of the ρ-VEX is done by combining processors with different issue widths, as can be seen in Fig. 2.

When using TMR with a common diverse processor strategy, the correct execution verification is always dependent of the slowest processor (i.e., faster processors would have to wait for the slowest one). However, in the case of the ISA-DTMR, the fastest processors can execute different applications while waiting for the slowest processor to finish, using this spare time to perform actual

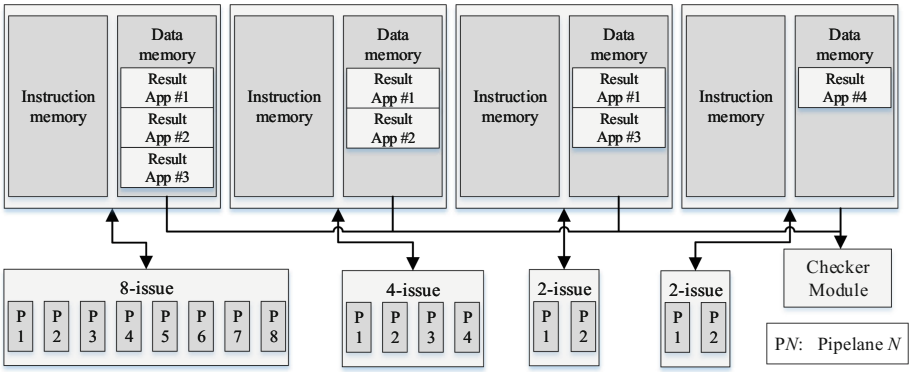


Fig. 2. Example of an ISA-DTMR implementation. A quad-core with one 8-issue core, one 4-issue and two 2-issue

computation [7]. If a processor executing an application with DMR protection fails, the voter module will flag the execution as faulty and trigger its re-execution so the fault can be corrected.

To keep the transparency as high as possible, the applications are only checked at the end of the execution by a specific hardware (Checker Module in Fig. 2). The Checker Module is responsible for verifying the correctness of the result, based on the reliability requirement of the given application. In case of a protected application, it compares the resulting memories to check the execution correctness. The Checker module implements a Finite-State Machine (FSM), which reads the application result from the memory in a burst, and the results are compared to detect errors. If there is a mismatch in a memory position, an error bit will flag the faulty result.

4 Results

In this section, we evaluate the proposed approach and analyze the results for fault tolerance, area, performance, power, and energy consumption.

4.1 Methodology

A total of nine designs were evaluated in this work: three homogeneous quad-core processors with issue-widths of 8 (baseline), 4, and 2 (i.e., 8-8-8-8, 4-4-4-4, 2-2-2-2); and six heterogeneous with the following configurations: 8-8-8-4, 8-8-4-4, 8-4-4-4, 8-4-4-2, 8-4-2-2, 8-2-2-2.

In order to analyze the reliability of the circuit, we have used a variation of the Mean Work to Failure metric, proposed in [15], as defined below.

$$MWTF = \frac{amount\ of\ work\ completed}{number\ of\ errors\ encountered} = \frac{core\ occupation}{(failure\ rate) \times (execution\ time)}$$

Where the core occupation is the ratio between the number of program instructions and the total number of instructions (program instructions plus NOPs). With that, it is possible to capture the trade-off between performance and fault tolerance and allows one to evaluate the reliability of different issue-widths removing the influence of the NOPs (free slots that the compiler was not able fill to exploit the instruction-level parallelism from the program) and the difference in execution time when the issue-width is reduced or increased.

To obtain the failure rate, a fault injection campaign was conducted and faults were injected at the gate-level signals of the design, using the Simbah-FI framework [17]. In this work, we consider that the memory and register file of the processors are protected with Error-Correcting Code (ECC). Our fault injection method simulates the occurrence of Single Event Transient (SET) faults, like the ones caused by radiation effects. The faults are injected at any internal and low-level signal of the target module. The injection instant follows a uniform probability function in the range between *zero* and t equal to the expected execution time from the application without faults, and to increase the likelihood of the SET to be captured by a flip-flop, the signal is forced for the duration of one clock cycle.

The fault injection campaign resulted in more than 500 thousand faults, injected at gate-level signals. For instance, the 8-issue homogeneous quad-core has 147,140 gate-level signals. To obtain accurate area and power measurements for ASICs, the designs were synthesized with the Cadence Encounter with a 65 nm CMOS cell library from STMicroelectronics. For the dynamic power consumption, the switching activity is considered to be of 30%, which is the traditionally assumed value for system level analysis of microprocessors [7]. When there is no switching activity in a given pipelane, only the static power dissipation is considered. For FPGAs, the design was synthesized to a Virtex-6 XC6VLX240T with the Xilinx ISE tool.

The applications used to evaluate our technique were selected from the ρ-VEX and the Powerstone benchmark suites [22] and they were divided into two scenarios that were tested in the nine different multicore designs:

- Scenario 1: FIR (TMR), POCSAG (DMR), LUDCMP (DMR), and CRC (unprotected execution).
- Scenario 2: Matrix multiplication (TMR), DFT (DMR), x264 (DMR), and Qurt (unprotected execution).

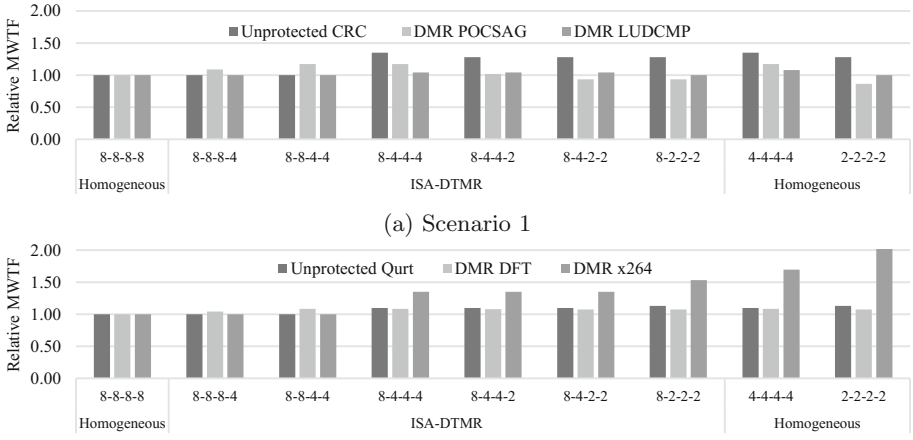
Even though the level of criticality of the selected applications may not reflect the same requirements they would have in a real system, they serve as examples for how ISA-DTMR behaves and to show the trade-offs when comparing it to conventional fault tolerance techniques. The applications were statically scheduled to the cores, aiming to minimize the total execution time as much as possible, for both the homogeneous and heterogeneous (ISA-DTMR) designs. In the heterogeneous designs, heavier applications are scheduled to the most powerful cores, following the same strategy as the schedulers used in ARM's big.LITTLE environments.

4.2 Fault Injection Campaign and MWTF Evaluation

As the FIR (*scenario 1*) and Matrix Multiplication (*scenario 2*) benchmarks are protected with TMR, they can mask all single faults in the design. By using ISA-DTMR, the replicates of these applications may be executed on cores with different configurations, exploiting heterogeneity with its advantages. The DMR applications (POCSAG and LUDCMP for *scenario 1*, and DFT and x264 for *scenario 2*) will result in re-execution when an error is detected, while the unprotected applications (CRC and Qurt) are considered non-critical and will remain with an incorrect result in case of a failure. Figure 3 presents the relative MWTF when compared to the 8-issue homogeneous quad-core processor (8-8-8-8), for the DMR-protected and unprotected applications. The applications with TMR have all their faults masked, thus it is not possible to evaluate their MWTF as the failure rate will be zero.

The amount of work that each application can perform until a failure occurs depends on the application's behavior and on which core configuration it is running on. For instance, when running the POCSAG application, the reduction in performance outweighs the reduction in the sensitive area, resulting in less work until a failure, compared to the 8-issue. On the other hand, for the x264 application (*scenario 2*), the 2-issue is able to improve the MWTF when compared to all other configurations. In this case, performance proportionally decreases less than the sensitive area, which means that even though the application takes longer to execute on the 2-issue processor, the processor is less exposed to radiation sources as its area is smaller.

This discussion highlights an important observation. By correctly choosing the core's issue-width in a heterogeneous processor, it is possible to improve the amount of work that an application can perform until a failure is detected in the system, even for those applications that are not critical.



(a) Scenario 1

(b) Scenario 2

Fig. 3. MWTF rate results for each scenario normalized to the 8-8-8-8 processor

4.3 Area Occupation and Power Dissipation

Table 1 presents the area (for both FPGA and ASIC) and power comparison (for ASICs) for the homogeneous designs and the proposed ISA-DTMR designs. The Checker Module occupies 1,581 cells and dissipates 0.89 mW of power for ASICs, and 426 LUTs (look-up tables) and 165 FF (flip-flop registers) for FPGAs. As expected, the ISA-DTMR implementation occupies less area compared to the homogeneous baseline implementation. In our designs, a 2-issue core occupies 40% less area compared to a 4-issue, and a 4-issue occupies 57% less area compared to an 8-issue. The proposed approach occupies considerably less area than the homogeneous 8-issue quad-core baseline design: the area reduction varies from 13.6% to 58.3% for FPGAs and from 14.2% to 54.9% for ASICs as the processor configuration is changed from 8-8-8-4 to 8-2-2-2. The power dissipation is also reduced from 14% (8-4-4-4) to 57.7% (8-2-2-2) when compared to the baseline. This will impact on energy consumption, as discussed next.

4.4 Energy Consumption and Performance

Figure 4 shows the results for energy consumption, area, and performance normalized to the homogeneous 8-issue processor. In scenario 1, by using ISA-DTMR with the 8-4-4-4 configuration, it is possible to reduce the energy consumption and area in 27.75% and 42.78%, respectively, while maintaining negligible performance overhead (less than 1%) when compared to the homogeneous 8-issue. In the 8-2-2-2 configuration, it is possible to reduce the energy and area by 29.86% and 55.21%, while having a performance overhead of 43.7%.

4.5 Combining Reliability, Energy Consumption, and Performance

As one could observe in the previous subsections, with ISA-DTMR, it is possible to trade-off the axes of performance, energy consumption, power dissipation,

Table 1. Power dissipation and area occupation

Design		ASIC		FPGA	
		Power (mW)	Cells	LUT	FF
Heterogeneous	2-2-2-2	34.4	70,493	31,262	10,773
	4-4-4-4	65.5	113,777	64,150	12,353
	8-8-8-8	149.6	262,705	140,362	15,853
ISA-DTMR	8-8-8-4	128.6	225,473	121,309	14,978
	8-8-4-4	107.5	188,241	102,256	14,103
	8-4-4-4	86.5	151,009	83,203	13,228
	8-4-4-2	78.8	140,188	74,981	12,833
	8-4-2-2	71.0	129,367	66,759	12,438
	8-2-2-2	63.2	118,546	58,537	12,043

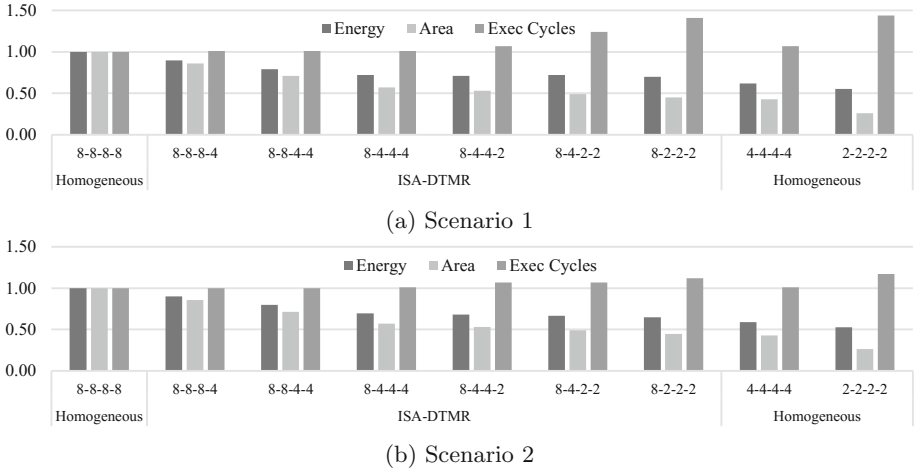


Fig. 4. Energy, area, and performance for each scenario normalized to the 8-8-8-8 processor

and area, providing expressive energy and area reductions at a low performance cost, and even improving the MWTF with the aforementioned savings in some cases. In this subsection, we summarize the benefits of applying the ISA-DTMR technique to improve fault tolerance in multicore systems.

Protecting applications with TMR, ISA-DTMR provides a fault tolerance mechanism to mask all single faults in the processor for critical applications and it still is able to reduce the area and energy consumption by executing the replicas on heterogeneous cores. Protecting applications with DMR, by using the proposed approach, it is also possible to improve the MWTF of the DMR-protected applications for almost all processor configurations, with the exception of the POCSAG application in the 8-4-2-2 and 8-2-2-2 processors, as discussed in Sect. 4.2. In the most significant case the MWTF is improved by more than 50% (x264 in the 8-2-2-2). This means that such applications will be able to perform more work until an error is detected and the re-execution of the application is performed, resulting in less re-executions. Therefore, the total energy consumption will be reduced as well as the performance overhead for all the re-executions. Executing Unprotected applications, even though these applications are not critical for the system, when the proposed technique is used, such applications are able to perform more work until a failure (up to 35%), when compared to the homogeneous 8-issue quad-core processor, which also means that these applications will fail less times. Thus, improving reliability just by choosing a core configuration that best fits the application behavior, instead of running all applications on homogeneous cores. In addition, the energy consumption can be reduced when choosing a smaller core without highly affecting performance for some applications, such as in the big.LITTLE approach.

5 Conclusion and Future Work

In this work, the ISA-DTMR is proposed, exploiting the fact that a number of different microarchitectures that implement the same ISA are available. The results showed that this technique can improve the fault tolerance consuming fewer resources, compared to the baseline, on both scenarios, with the exception of the 8-4-2-2 and 8-2-2-2 designs. The 8-4-4-2 exhibit the most balanced configuration, improving fault tolerance with more than 28% of reduction in energy consumption and half area occupation, compared to the baseline.

This work also shows that it is possible to improve the applications reliability by correctly choosing the core's issue-width. As future work, this technique will be applied to other processor architectures and more scenarios with different applications will be assessed. In addition, a dynamic scheduler and dynamic application criticality assessment mechanism will be implemented, and a software-based checker will be assessed and compared to the hardware approach.

Acknowledgement. This work was supported in part by CNPq, FAPERGS, and CAPES.

References

1. Arm Limited: Arm DynamIQ technology framework to design and build Cortex-A CPU systems (2017). https://developer.arm.com/technologies/dynamiq
2. Ashraf, R.A., Mouri, O., Jadaa, R., Demara, R.F.: Design-for-diversity for improved fault-tolerance of TMR systems on FPGAs. In: 2011 International Conference on Reconfigurable Computing and FPGAs, pp. 99–104, November 2011
3. Avizienis, A., Kelly, J.P.J.: Fault tolerance by design diversity: concepts and experiments. Computer **17**(8), 67–80 (1984)
4. Beck, A.C.S., Lisbôa, C.A.L., Carro, L.: Adaptable Embedded Systems. Springer Science & Business Media, Heidelberg (2012). https://doi.org/10.1007/978-1-4614-1746-0
5. Bolchini, C.: A software methodology for detecting hardware faults in VLIW data paths. IEEE Trans. Reliab. **52**(4), 458–468 (2003)
6. Bolchini, C., Carminati, M., Miele, A.: Self-adaptive fault tolerance in multi-/many-core systems. J. Electron. Test. **29**(2), 159–175 (2013)
7. Geuskens, B., Rose, K.: Modeling Microprocessor Performance. Springer Science & Business Media, Heidelberg (2012). https://doi.org/10.1007/978-1-4615-5561-2
8. Kriebel, F., Rehman, S., Sun, D., Shafique, M., Henkel, J.: ASER: adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era. In: ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6, June 2014
9. Kriebel, F., Shafique, M., Rehman, S., Henkel, J., Garg, S.: Variability and reliability awareness in the age of dark silicon. IEEE Des. Test **33**(2), 59–67 (2016)
10. Littlewood, B.: The impact of diversity upon common mode failures. Reliab. Eng. Syst. Saf. **51**(1), 101–113 (1996)
11. Mukherjee, S.S., Kontz, M., Reinhardt, S.K.: Detailed design and evaluation of redundant multi-threading alternatives. In: Proceedings 29th Annual International Symposium on Computer Architecture, pp. 99–110 (2002)

12. Pillai, A., Zhang, W., Kagaris, D.: Detecting VLIW hard errors cost-effectively through a software-based approach. In: 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW 2007, vol. 1, pp. 811–815, May 2007
13. Ray, J., Hoe, J.C., Falsafi, B.: Dual use of superscalar datapath for transient-fault detection and recovery. In: MICRO-34 Proceedings of the 34th ACM/IEEE International Symposium on Microarchitecture, pp. 214–224, December 2001
14. Reinhardt, S.K., Mukherjee, S.S.: Transient fault detection via simultaneous multithreading. In: Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No. RS00201), pp. 25–36, June 2000
15. Reis, G.A., Chang, J., Vachharajani, N., Mukherjee, S.S., Rangan, R., August, D.I.: Design and evaluation of hybrid fault-detection systems. In: 32nd International Symposium on Computer Architecture (ISCA 2005), pp. 148–159, June 2005
16. Sabena, D., Reorda, M.S., Sterpone, L.: On the development of software-based self-test methods for VLIW processors. In: IEEE Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 25–30, October 2012
17. Sartor, A.L., Becker, P.H.E., Beck, A.C.S.: Simbah-FI: simulation-based hybrid fault injector. In: 2017 VII Brazilian Symposium on Computing Systems Engineering (SBESC), pp. 94–101, November 2017
18. Sartor, A.L., Becker, P.H.E., Hoozemans, J., Wong, S., Beck, A.C.S.: Dynamic trade-off among fault tolerance, energy consumption, and performance on a multiple-issue VLIW processor. IEEE Trans. Multi-scale Comput. Syst. **55**(99), 1 (2017)
19. Sartor, A.L., Lorenzon, A.F., Carro, L., Kastensmidt, F., Wong, S., Beck, A.C.S.: A novel phase-based low overhead fault tolerance approach for VLIW processors. In: Computer Society Annual Symposium on VLSI, pp. 485–490, July 2015
20. Sartor, A.L., Wong, S., Beck, A.C.S.: Adaptive ILP control to increase fault tolerance for VLIW processors. In: Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 9–16, July 2016
21. Sartor, A.L., Lorenzon, A.F., Carro, L., Kastensmidt, F., Wong, S., Beck, A.C.S.: Exploiting idle hardware to provide low overhead fault tolerance for VLIW processors. J. Emerg. Technol. Comput. Syst. **13**(2), 13:1–13:21 (2017)
22. Scott, J., et al.: Designing the low-power m* core architecture. In: IEEE Power Driven Microarchitecture Workshop. Citeseer (1998)
23. Shye, A., Moseley, T., Reddi, V.J., Blomstedt, J., Connors, D.A.: Using process-level redundancy to exploit multiple cores for transient fault tolerance. In: IEEE/IFIP Conference on Dependable Systems and Networks, pp. 297–306, June 2007
24. Sterpone, L., Sabena, D., Campagna, S., Reorda, M.S.: Fault injection analysis of transient faults in clustered VLIW processors. In: IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 207–212, April 2011
25. Tambara, L.A., Kastensmidt, F.L., Azambuja, J.R., Chielle, E., Almeida, F., Nazar, G., Rech, P., Frost, C., Lubaszewski, M.S.: Evaluating the effectiveness of a diversity TMR scheme under neutrons. In: European Conference on Radiation and its Effects on Components and Systems (RADECS), pp. 1–5, September 2013
26. Wang, Z., Yang, L., Chattopadhyay, A.: Architectural reliability estimation using design diversity. In: Symposium on Quality Electronic Design, pp. 112–117, March 2015
27. Wong, S., van As, T., Brown, G.: ρ-VEX: A reconfigurable and extensible softcore VLIW processor. In: Conference on Field-Programmable Technology, pp. 369–372, December 2008