Reducing Overprovision of Triple Modular Reduncancy Owing to Approximate Computing

Bastien Deveautour, Marcello Traiola, Arnaud Virazel, Patrick Girard

Reducing Overprovision of Triple Modular Reduncancy Owing to Approximate Computing

Bastien Deveautour [1] Marcello Traiola [2] Arnaud Virazel [1] Patrick Girard [1]
[1] LIRMM, Univ. of Montpellier / CNRS - <lastname>@lirmm.fr
[2] INL, UMR5270, ECL, Univ Lyon, INSA Lyon, CNRS, UCBL, CPE Lyon - marcello.traiola@ec-lyon.fr

***Abstract*—Until recently, Approximate Computing (AxC) was considered to be a trend topic mainly for resilient applications. Its use aimed at reducing area and power consumption of Integrated Circuits (ICs) at the cost of a reduced accuracy. Beside, AxC-based fault-tolerance has also emerged recently to save area et power consumption w.r.t. conventional fault-tolerance at the cost of a reduced reliability level. Therefore, approximate fault-tolerance is restricted to non-critical applications. In a previous work, a Quadruple Approximate Modular Redundancy (QAMR) scheme, based on using four approximate circuit copies, was proposed. In a single fault scenario, it provides the same reliability level as Triple Modular Redundancy (TMR). Moreover, QAMR allows reducing area and power consumption costs w.r.t. TMR in many cases. In this paper, we study the occurrence of multiple faults, and compare the QAMR and TMR behaviors. Experimental results highlight the TMR overprovision (i.e. it provides more redundancy than necessary) and show how the QAMR approach can reduce it. Moreover, a thorough analysis of the results shows that a high tolerance to multiple faults is associated to a large circuit area and to a lower percentage of logic shared among different fan-in cones of the circuit outputs.**

***Index Terms*—Approximate computing; fault tolerance; modular redundancy; fault injection.**

I. Introduction

During its lifespan, the hardware of a system used in harsh (e.g. radiative) environment is subject to various physical phenomena that may alter its performance or provoke errors [1]. Moreover, shrinking dimensions of transistors and increasing density of integration exacerbate aging effects that may lead to permanent faults as well as transient faults induced by energetic charged particles [2]. When those faults propagate through the logic, they can be captured by memory cells (e.g. flip-flops) and stored as faulty values. In that case, captured transient and permanent faults lead to soft and hard errors respectively. Both errors may cause a system malfunction.

Error significance depends on the application running on the system. Some applications are resilient to faults (e.g. speech recognition, image encoding, etc.) and can continue to operate correctly even when some errors occur. This interesting resiliency property has inspired the AxC paradigm. AxC utilizes inaccurate computations, rather than accurate, to achieve gains in performance/circuit area/power consumption at the cost of a reduced accuracy [3]. *AxC* has been applied to resilient applications where an approximate result is sufficient for their purpose [4]. From the hardware standpoint, *AxC* enables the creation of extremely efficient *Approximate Integrated Circuits (AxICs)* whose output values may differ from the original for a certain set of input values [5].

Unlike resilient applications, safety-critical applications cannot tolerate errors as they may lead to risks for human lives or very high costs in damage control. To alleviate this issue, several fault tolerance schemes have been designed to maintain a guaranteed level of reliability. A well-known existing scheme capable of tolerating soft and hard single errors is the *Triple Modular Redundancy* (*TMR*) [6]. TMR is a fault-tolerant scheme composed of three identical instances of a circuit that are connected to a majority voter.

The fault tolerance of TMR has been estimated from several perspectives. In [7], the yield and product quality of TMR are studied, and authors state that TMR can tolerate 100% of single faults occurring in one of its three modules. They also show that TMR is able to tolerate multiple faults when different modules incur faulty values on different outputs. In [8], authors give a classification of three different types of events that may occur in TMR when the voter in unable to mask faults occurring in different modules. Faults are classified as: *type_1)* all input lines of the voter have the same incorrect logic value; *type_2)* two of the three input lines of the voter have the same incorrect logic value; *type_3)* all input lines of the voter exhibit different incorrect logic values. According to this classification, the probability for each type of faults is the product of the probabilities of each voter input line to have a fault propagated to them. In [9], authors performed single and double fault injection experiments on a scheme called *Multiple Error Recovery Technique for Triple Modular Redundancy Systems* (*SMERTMR*). This modified TMR can locate and correct 99.7% of multiple permanent faults affecting two modules.

The counterpart of the high tolerance level offered by the TMR is a considerable cost in terms of area and power overhead compared to the initial area of the unprotected circuit – about 200%. Until recently, no other scheme has been proposed to achieve a similar tolerance level for both transient and permanent faults at lower cost. In [10] and [11], authors show that using approximate schemes to create an *Approximate Triple Modular Redundancy* (*ATMR*) is possible. As TMR, ATMR is composed of three instances of a circuit. However, only one instance is a precise version of the initial circuit whereas the others are approximate versions that give at least one precise output for the majority vote to be correct. These modifications to the TMR lower the area costs but come at the expense of a reduced error-masking capability, which makes ATMR not

suitable in safety-critical scenarios.

In [12], we proposed a *Quadruple Approximate Modular Redundancy* (*QAMR*) architecture to overcome the above issue while maintaining low area costs. QAMR ensures a full logic masking (tolerance) of transient and permanent faults by using four approximate circuit replicas. The required fundamental condition is that, at a given time, at least three out of four instances of the QAMR provide a precise (i.e., non-approximated) response. The majority voter is the same as for TMR, and guarantees that QAMR achieves the same tolerance level as TMR in case a single fault occurs during operation of the circuit.

The motivation of this work lies in the observation that a TMR structure tolerates 100% of single faults and it is able to tolerate multiple faults in some cases [7]. However, multiple fault tolerance is not the TMR goal. Moreover, there is *no guarantee* that multiple faults will be tolerated (as it depends on where faults in TMR modules will appear). Therefore, we argue that the TMR structure is *overprovisioned* w.r.t. single fault tolerance, i.e. it provides more redundancy than necessary, thus inducing more area overhead than necessary. Based on this observation, we infer that TMR multiple fault tolerance can be sacrificed to reduce the area overhead.

In this work, we show how QAMR allows reducing the TMR overprovision (and hence area overhead) while still guaranteeing a complete single fault tolerance. We report results of multiple fault injection campaigns realized on both TMR and QAMR schemes. The goal is to analyze the impact of double, triple and quadruple faults occurring in both schemes, and to compare their respective fault tolerance level in each scenario. Results show that the QAMR ability to reduce TMR area overhead stems from a reduction of its overprovision (i.e. multiple fault tolerance). A deeper analysis of the results shows that the fault tolerance has a strong dependence on the circuit internal structure. Thus, using AxC to modify the internal structure of the circuit replicas is a promising approach to reduce the TMR area overprovision (i.e.: reduce area overhead). In particular, QAMR achieves this goal while unaltering its overall single-fault tolerance abilities.

The remainder of the paper is organized as follows. Section II reviews our previous work on the QAMR architecture. Section III presents the fault injection methodology. Section IV reports the results achieved and their implications. Conclusion and future directions for this work are given in Section V.

II. Previous Works

In [12], we introduced a new scheme whose goal is to achieve the TMR single fault tolerance level while reducing area cost. As for TMR, the goal is to protect the whole circuit against 100% of the permanent and transient single faults.

QAMR is built as follows. For a given multi-output Boolean function, (i) we randomly select four disjoint subsets of its outputs (S_1, S_2, S_3, S_4), (ii) we instantiate four copies of the function (C_1,C_2,C_3,C_4) (iii) we remove, from each C_i, a different subset S_i (and its corresponding fan-in logic), and (iv) we perform a logic synthesis. In this way, we obtain four AxICs ($AxIC_1$, $AxIC_2$, $AxIC_3$, $AxIC_4$) whose outputs are connected to a conventional majority voter (i.e. the same as used in TMR). Then we can compare the area cost of QAMR with respect to TMR. TMR is realized by triplicating the Boolean function and performing the same logic synthesis as in step (iv). We iterate the above process to find a cheaper QAMR implementation in terms of area with respect to TMR. For more details, the reader can refer to [12]. Figure 1 illustrates a simplified example of a QAMR implementation for a 4-bit output circuit. A different

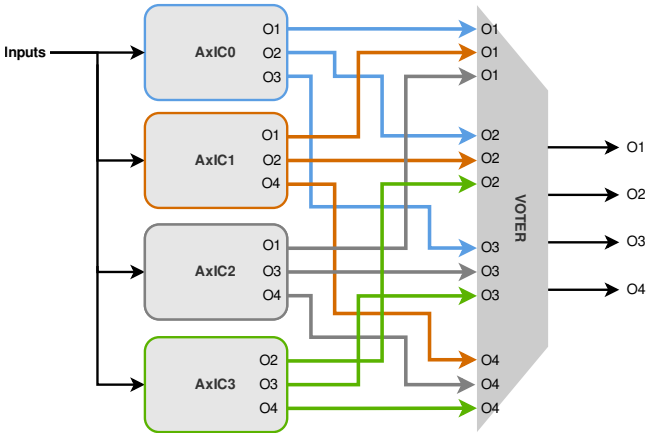


Fig. 1: QAMR scheme

group of logic cones composes each AxIC. This allows the synthesis process to perform different optimizations. Moreover, the use of a conventional majority voter is possible, as in TMR, since the triplication of each output is still the case in QAMR.

In [12], we compared TMR and QAMR by using the *Relative Area Gain* (*RAG*) metric expressed as follows:

$$RAG = \frac{3 \cdot A_P - (A_{X1} + A_{X2} + A_{X3} + A_{X4})}{3 \cdot A_P} \quad (1)$$

where *A*p represents the area of the triplicated precise circuit in the TMR. Axi represents the area of the approximate module i in the QAMR. Note that a negative RAG indicates that the TMR implementation has a lower area cost than its QAMR equivalent.

Figure 2 summarizes results presented in [12] on the LGSynth'91 combinational benchmark circuits with five or more outputs. Results indicate that 19 out of 52 circuits have a positive *RAG* when using *QAMR* scheme. Some circuits like *Apex1* or *K2* have a *RAG* higher than 20%. Conversely, circuits with a negative *RAG* stay above -10%. Peculiarly, the only two circuits with a negative *RAG* below -10% are decoders and their approximate modules do not offer any area benefit since each removed output only leads to the logic removal of one single gate. As stated in [12], these results indicate that QAMR can be an effective solution to reduce the TMR area overhead and highlight the need of a new approximation process for QAMR to achieve better results.

Regarding the fault tolerance of the two architectures, in case of a single fault both QAMR and TMR provide 100% fault tolerance for any input vector. In this work, we show that the QAMR reduced area overhead is associated to a reduced overprovision (i.e. multiple fault tolerance).

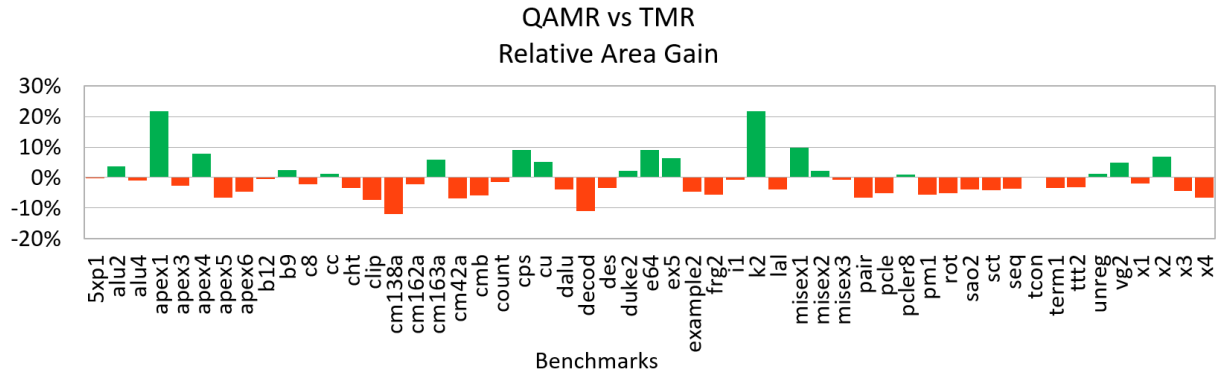


Fig. 2: Relative Area gain of the QAMR with respect to the TMR

III. Multiple Fault Injection Methodology

To assess and compare the multiple fault tolerance of the TMR and QAMR, we performed simulation-based fault-injection experiments by using our ad-hoc fully automated gate-level fault-injection framework presented in [14]. Gate-level simulation is suitable to perform fault-injection experiments as, unlike micro architectural-level simulation, it accurately models most of the physical defects and transient faults and it is much faster than transistor-level simulation.

```
if TMR then modules = {PC0,PC1,PC2}
else if QAMR then modules = {AxC0,AxC1,AxC2,AxC3};
switch (injection) {
  case : "double":
    //inject 1 random fault per selected modules
    2_fault_sites = rand.pick_2 (modules);
    break;
  case : "triple":
    //inject 1 random fault per selected modules
    3_fault_sites = rand.pick_3 (modules);
    break;
  case : "quadruple":
    //4 random faults within at least 3 modules
    4_fault_sites = rand.pick_3 (modules);
    4_fault_sites += rand.pick_1 (modules);
    break;}
```

Fig. 3: Fault injection algorithm

Fault injection campaigns have been carried out in such a way that multiple faults need to affect at least two modules of the QAMR. Figure 3 depicts the multiple fault injection algorithm for each scenario. Each module of the TMR and QAMR has its specific list of fault sites. For $i \in \{2, 3, 4\}$, the *i_fault_sites* variables contain *i* different fault sites selected from the fault list of each module of the TMR or QAMR. To this end, *rand_pick_i* selects *i* random modules. In each module, it selects a random fault site. For instance, let us assume that in a double injection case, rand.pick_2 selects the two precise replicas, PC0 and PC1, of the given circuit. The algorithm will then randomly select one fault site for PC0 and another for PC1.

The above protocol ensures that a double fault injection cannot have the same impact as a single fault injection; a triple fault injection cannot have the same impact than a single or double fault injection; and a quadruple fault injection cannot have the same impact than a single, double or triple fault injection. Note that we do not consider the majority voter as a fault injection target since it is equivalently used by QAMR and TMR, so that any fault occurrence in it does not entail any fault tolerance differences.

We conducted experiments on five of the Combinational Multi-Level and Two-Level circuits from the publicly available LGSynth'91 benchmark suite [13]. In particular, we selected these circuits for the variety of RAG values obtained in [12] and for their different size, as shown later in Table I. For circuit synthesis, we used Design Compiler from Synopsys [15] and the NanGate 45nm Open Cell Library [16]. We obtained the fault injection list and test benches by using Synopsys TetraMAX [15]. Finally, we performed the simulation-based fault injections with ModelSim from Mentor Graphics [17].

The number of injected transient faults was defined by the approach proposed in [18] with a 1% margin of error and a 95% of confidence. For each multiple fault injection campaigns, the number of injected faults corresponds to the number of cycles in which we inject two, three or four transient faults accordingly. In this way, the number of injected faults is large enough to achieve a significant distribution between masked faults and faults leading to wrong outputs.

Transient faults are modeled as digital pulses using three parameters: fault location *l*, fault-injection time *t* and duration *d* representing a *Single Event Transient* (*SET*) pulse width. The pulse width of particle-induced SETs varies depending on factors like type of radiations, capacitance of the impacted net and process technology [19]. We randomly selected pulse widths in a range between 0.25ns and 1.25ns. The values of this range were chosen in accordance to the typically anticipated SET pulse widths in 45nm technology. We select parameters *l* and *t* randomly at each new injection. We perform the injection by flipping the signal value at the *l* location, at the time *t*. Also, all faults in a given injection have the same values for parameters *t* and *d* (i.e. in case of a double fault injection, both faults f_1 and f_2 are activated at the same time $t_1 = t_2$ and for the same duration $d_1 = d_2$, but with $t_1 \neq d_1$). Note that,

we added a delay on the circuit outputs so that the majority voter can complete its computation before the capture is done. As workload, we used a test sequence detecting all possible stuck-at faults. The test set was generated so that each fault is detected by 50 test patterns (i.e., ATPG *N-detect* option set to 50). This allows the fault injection scenarios to benefit from a wide variety of input vectors. As already explained, the majority voter structure is excluded from the injection campaigns.

Table I lists the features of the various circuits used for these experiments: number of inputs and outputs, *RAG*, number of gates and number of injection cycles for each campaign. One can notice that the selected circuits have very different profiles. For instance, for *clip* circuit the QAMR area cost is higher compared to its TMR counterpart (negative RAG). Conversely, *e64* and *k2* have a large number of gates and a QAMR implementation that has lower area overhead (positive RAG) than the corresponding TMR version. For the two other circuits, *alu4* and *b12,* TMR and QAMR have roughly the same area (RAG close to 0%). However, both *b12* and *alu4* have a significant gate count difference.

TABLE I: Case Study Specification

Circuit		IN	OUT	RAG	# Gates	# Injection cycles
b12	TMR	15	9	-0.44%	160	486
	QAMR				163	477
clip	TMR	9	5	-7.36%	285	1057
	QAMR				311	1134
alu4	TMR	14	8	-1.5%	1713	4770
	QAMR				1789	4862
k2	TMR	45	45	21.81%	3515	6712
	QAMR				2918	6029
e64	TMR	65	65	8.66%	1131	2855
	QAMR				1068	2584

Since we perform the fault injection by simulating the gate-level models of the QAMR and TMR architectures, we have access to all nets in the circuits. Consequently, we can compare the voter outputs with their corresponding golden responses. After the fault injection, we classify faults into two categories:

- *Silent faults:* faults having no effect on the results. These faults are filtered/masked thanks to the redundancy provided by the architecture. The fault masking effect stems from the circuit's logic intrinsic resiliency and/or the majority voter, which delivers the correct output when at least two replicas produce correct values.
- *Fail-silent faults*: faults leading to erroneous results. During workload application, at least two replicas deliver a corrupted value for the same output net. In this case, the majority voter propagates the corrupted value to its output and the faults remain undetected. These faults are the most critical ones as the result provided is wrong without any error indication.

IV. Experimental Results

Figures 4, 5, and 6 show the results of the multiple fault injection campaigns on b12, clip, and alu4, respectively. These figures report the obtained fail-silent fault rate (y-axis) of a

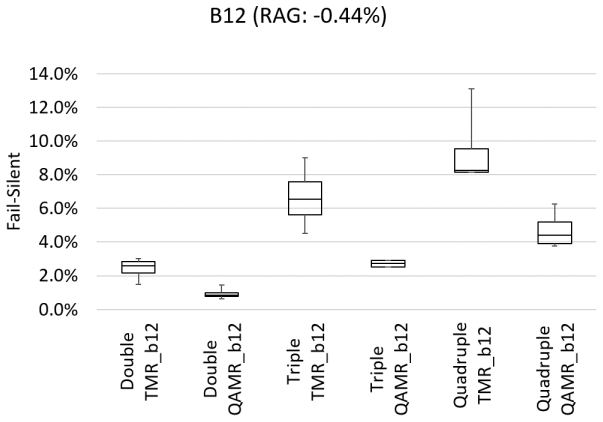


Fig. 4: b12 multiple fault injection

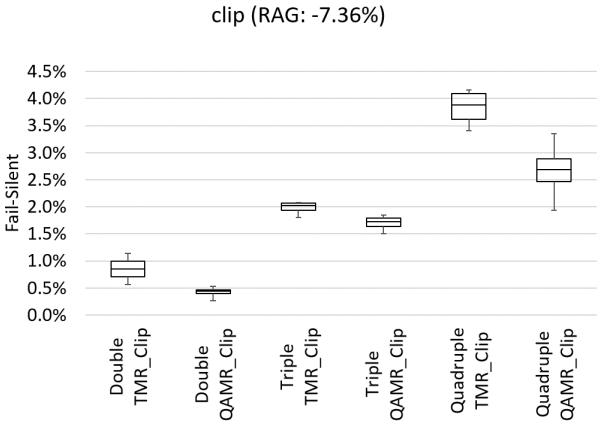


Fig. 5: clip multiple fault injection

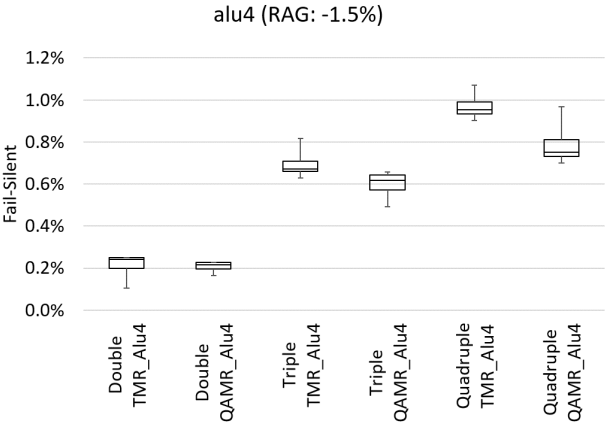


Fig. 6: alu4 multiple fault injection

double, triple and quadruple fault injection (x-axis) in both TMR and QAMR. Note that the silent fault rate is not shown but can be calculated as $100\% - \%$ of fail silent faults. We observe the highest average fail-silent fault rate for b12 (Figure 4), achieving 8.3% (TMR) and 4.4% (QAMR) in the quadruple fault injection scenario. The lowest average fail-silent fault rate is observed for alu4 (Figure 6), achieving 0.24% (TMR) and 0.21% (QAMR) in the double fault injection scenario. In general, as expected, injecting more faults leads to an increasing rate of fail-silent faults in both schemes. These results show that the QAMR is more resilient to multiple faults and have a higher area overhead than its TMR counterpart (i.e.: the higher QAMR fault tolerance stems from its overprovision).

Coherently, the fail-silent fault rate of *k2* and *e64* circuits (reported in Figures 7 and 8 respectively) show that QAMR scheme is less resilient to multiple faults than TMR (for every injection scenario) whereas QAMR has a reduced area cost compared to the TMR (i.e., positive RAG). For instance, the

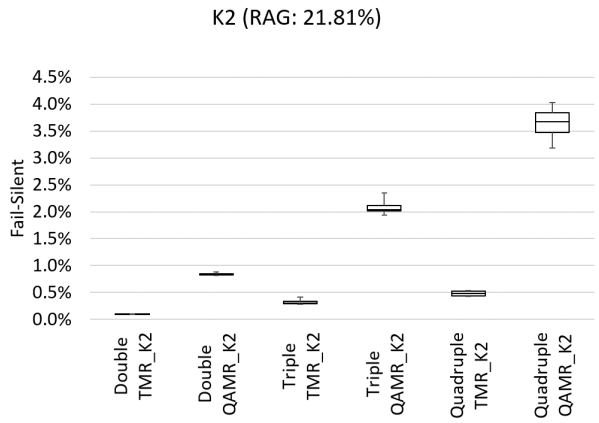


Fig. 7: k2 multiple fault injection

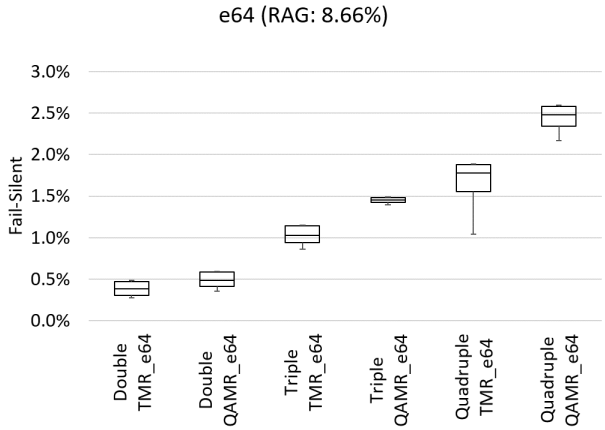


Fig. 8: e64 multiple fault injection

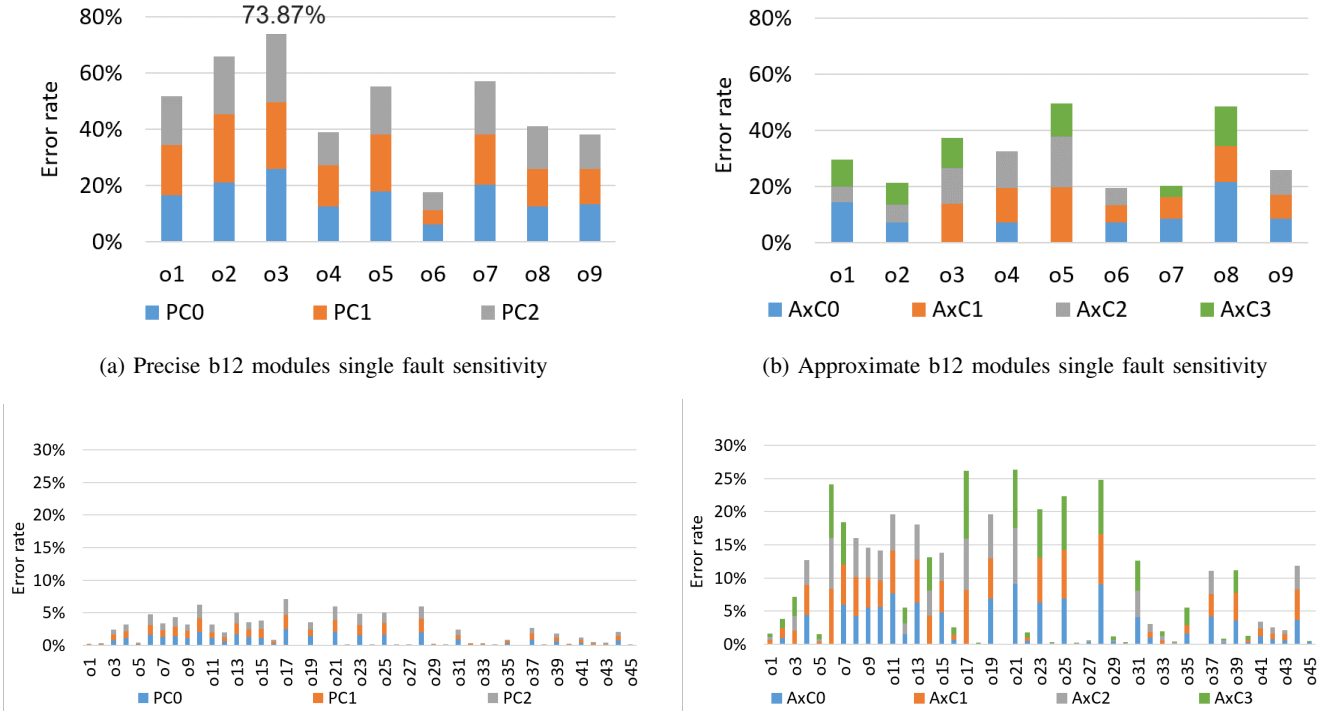
TMR version of *K2* has only 0.1% of fail-silent faults in the double-fault injection scenario and its fail-silent fault rate in a quadruple fault injection scenario barely increases (by 0.35%) with respect to the double fault scenario.

By keeping in mind that **QAMR still guarantees 100% single fault masking**, these results highlight that the capability of QAMR to reduce the area overhead stems from the reduction of the overprovision entailed by the TMR approach. In other words – since multiple fault resilience *is not* the TMR goal – QAMR successfully manages to provide 100% single fault masking and reduced area overhead by *reducing the multiple fault resilience*.

Let us now analyze the results more thoroughly. We observe that the smaller circuit, *b12*, has the highest fail-silent fault rate. On the contrary, the biggest circuits have the lowest fail-silent fault rate. For instance, the double injection scenario for *b12* shows a fail-silent fault rate – for both QAMR and TMR – around ten times higher than the rates shown in the double injection scenario for *alu4*. Based on these observations, we can infer that a circuit with a larger number of logical gates has a better masking effect than circuits with fewer logical gates. Results in Figures from 4 to 8 are aligned with this, i.e. bigger circuits are less sensitive to faults.

To further corroborate this statement, we want to observe the masking effect difference in a single fault injection scenario within each module of the TMR and the QAMR schemes. Figure 9 shows the error rate (y-axis) of a single fault injection within the *Precise Circuit (PC1, PC2 and PC3)* modules of *b12* composing the TMR (Figure 9.a), and the same results on approximate modules *(AxC1, AxC2, AxC3 and AxC4)* of *b12* composing the QAMR (Figure 9.b). Similar results are provided for circuit *k2* in Figures 9.c and 9.d. We performed the fault injection in each module – precise and approximate – independently. The x-axes report the name of the initial circuit outputs. Each bar in the graph stacks the error rate values of the three replicas' outputs. Each color corresponds to a different replica. As can be seen, *b12* approximate modules exhibit a lower error rate (i.e. higher masking effect) than the corresponding precise modules. For instance, for the output *o3*, the three precise copies achieve, together, an error rate of nearly 75% (Figures 9.a). On the other hand, the approximate counterparts (Figure 9.b) reach, together, less than 40% error rate. This is in line with the above statement, since the QAMR implementation for b12 has a negative RAG. On the contrary, *k2* precise modules have a higher masking effect than the approximate counterparts. In particular, the error rate for the precise modules' outputs barely reach 8%, while many of the approximate outputs exceed 20% error rate. Again, this is in line with the above statement, since the QAMR implementation for *k2* has a positive RAG.

Results in Figure 9 suggest that the masking effect might depend *not only* on the circuit area. For instance, *b12* error rate in the approximate modules is lower than in the precise ones. However, their area difference is almost negligible RAG -0.44%). Moreover, in the case of *k2*, the bigger area difference (RAG 21.81%) hardly explains alone the difference of a factor five in error rate between modules in both TMR and QAMR for a single fault injection scenario. Furthermore, we observe that a single fault in an approximate module of *k2* tends to impact



(a) Precise b12 modules single fault sensitivity

(b) Approximate b12 modules single fault sensitivity

(c) Precise k2 modules single fault sensitivity

(d) Approximate k2 modules single fault sensitivity

Fig. 9: Single fault analysis of the modules composing the TMR and QAMR structures of circuits b12 (a and b) and k2 (c and d)

a greater number of outputs than a single fault on the precise modules. This fact suggests that the approximate modules of *k2* have more logic gates shared by multiple outputs than its precise version. This leads us to a further analysis.

In our previous work [12], we proposed to measure the logic leading to more than one output as *Shared Logic Rate (SLR)*. In a circuit, SLR corresponds to the number of nodes that lead to more than one output divided by the total number of nodes.

TABLE II: SLR of Different b12 Modules

b12 module	PC	AxIC0	AxIC1	AxIC2	AxIC3
SLR	16.8%	23.2%	19.6%	19%	19%

TABLE III: SLR of Different k2 Modules

k2 module	PC	AxIC0	AxIC1	AxIC2	AxIC3
SLR	16.2%	54.8%	55.2%	52.4%	53.8%

Tables II and III show the SLR of the precise and approximate modules composing *b12* and *k2* respectively in their TMR and QAMR schemes. The tables show that the approximate modules (AxIC0 to AxIC3) of *b12* have a low SLR, i.e. around 20%, similar to the precise one (PC) ∼17%. Conversely, the approximate modules of *k2* have an SLR higher than 50%, quite far from 16.2% SLR of the precise *k2* circuit.

Based on these results, we can infer that the fault tolerance of TMR and QAMR for the multiple fault injection scenarios depends on two factors: the area and the SLR of their composing modules. Indeed, the larger the number of gates, the greater the fault masking effect; the greater the SLR, the higher the number of output logic cones towards which a fault propagates.

V. Conclusion

In this paper, we showed how approximate computing paradigm enables the area overhead reduction of a TMR architecture, while not altering the overall single fault tolerance. Indeed, we argue that the TMR area overhead is due to overprovisioning and we showed how the QAMR approach [12] successfully provides 100% single fault masking and reduces the TMR area overhead by reducing its multiple fault resilience. In fact, the multiple fault resilience is not the TMR goal, thus it can be sacrificed to reduce the area overhead. Our results show that a higher tolerance to multiple faults is associated to a larger circuit area and to a lower percentage of logic shared among different fan-in cones of the circuit outputs. As future work, we aim at fully exploiting optimization opportunities provided by approximate computing to realize enhanced approximation techniques oriented towards extremely optimized QAMR implementations.

VI. REFERENCES

1. K. Weide-Zaage and M. Chrzanowska-Jeske, "Semiconductor Devices in Harsh Conditions," CRC Press, 2016, ISBN: 9781315368948
2. M. Sachdev, "Defect Oriented Testing for CMOS Analog and Digital Circuits," Springer, 2013, ISBN 9781475749267
3. Q. Xu, T. Mytkowicz and N. S. Kim, "Approximate Computing: A Survey, " IEEE Design & Test, vol. 33, no. 1, pp. 8-22, Feb. 2016.
4. A. Sanchez-Clemente, L. Entrena, M. Garcia-Valderaz and C. Lopez-Ongil, "Logic masking for SET mitigation using approximate logic circuits," *Proc. of* IEEE Int'l On-Line Testing Symp, pp. 176-181, *2012*.
5. S. Mittal, "A survey of techniques for approximate computing," *ACM Computer Survey*, vol. 48, no. 4, pp. 62:1–62:33, March 2016.
6. R. E. Lyons, W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability" *IBM Journal of Research and Development*, vol. 6, N° 2, April 1962.
7. M. Hunger and S. Hellebrand, "The Impact of Manufacturing Defects on the Fault Tolerance of TMR-Systems," *IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 101-108, Nov. 2010.
8. S. R. Nanduri, A. K. El Hakeem and A. J. Al-Khalili, "Fault analysis of a TMR system using multiple valued logic, " *Annual International Phoenix Conference on Computers and Communications*, pp. 23-29, March 1990.
9. F. A. Siddiqui and P. Gour, "Scan-chain-based multiple error recovery in TMR systems (SMERTMR), " *Innovative Applications of Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH)*, pp. 374-378, Nov. 2014
10. Iuri A. C. Gomes, M. Martins, A. Reis and F. Lima Kastensmidt, "Using only redundant modules with approximate logic to reduce drastically area overhead in TMR," *in Proc. IEEE Latin-American Test Symp.*, pp. 1–6, March 2015.
11. Iuri A.C. Gomes, M. Martins, A. Reis, F. Lima Kastensmidt, "Exploring the use of approximate TMR to mask transient faults in logicwith low area overhead," *Microelectronics Reliability*, vol 55, no 9–10, pp 2072-2076, Aug.-Sept. 2015.
12. B. Deveautour, M. Traiola, A. Virazel and P. Girard, "QAMR: an Approximation-Based Fully Reliable TMR Alternative for Area Overhead Reduction," *in Proc.* IEEE European Test Symp., pp. 1-6, May 2020.
13. S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Technical Report 1991-IWLS-UG-Saeyang, MCNC.
14. I. Wali, A. Virazel, A. Bosio, P. Girard, S. Pravossoudovitch, M. Sonza Reorda, "A Hybrid Fault-Tolerant Architecture for Highly Reliable Processing Cores", *Journal of Electronic Testing – Theory and Applications*, vol. 32, no. 2, pp. 147–161, March 2016.
15. Design Compiler & TetraMAX [Online]. Available: https://www.synopsys.com/
16. NanGate. Nangate 45nm open cell library. [Online]. Available: http://www.nangate.com/?page id=2325.
17. ModelSim. [online]. Available: https://eda.sw.siemens.com/en-US/ic/modelsim/
18. R. Leveugle, A. Calvez, P. Maistri, and P. Vanhauwaert, "Statistical fault injection: Quantified error and confidence", *in Proc. of IEEE/ACM/EDAA Design Automation and Test in Europe*, pp. 502–506, April 2009.
19. G. Wirth, Kastensmidt, L. Fernanda, I. Ribeiro, "Single Event Transients in Logic Circuits-Load and Propagation Induced Pulse Broadening," *IEEE Trans. on Nuclear Science*, vol. 55, no. 6, pp. 2928-2935, Dec. 2008.