Design Space Exploration of an Approximation-Based Fully Reliable TMR Alternative

Marcello Traiola[1], Jorge Echavarria[2],
Alberto Bosio[1], Jürgen Teich[2] and Ian O'Connor[1]
[1]INL, École Centrale de Lyon
[2]Department of Computer Science, Friedrich-Alexander-Universität (FAU)
[1]Lyon, France – [2]Erlangen-Nürnberg, Germany

[1]firstname.lastname@ec-lyon.fr - [2]firstname.lastname@fau.de

Abstract. In the last decade, Approximate Computing (AxC) has been studied as a possible alternative computing paradigm. Its adoption has been particularly beneficial especially in the field of error tolerant applications. In this context, AxC has been recently used to reduce the costs related to conventional fault tolerant schemes, such as the Triple Modular Redundancy (TMR). Unfortunately, existing approximate solutions cannot ensure the same fault-tolerance capability as the conventional TMR. In a previous work, we proposed the Quadruple Approximate Modular Redundancy (QAMR), a novel approximation-based fault-torerant solution. QAMR can reduce the cost compared to conventional TMR structures, while guaranteeing the same fault-tolerance capability. In this article, we adopt a new approximation technique to realize the QAMR and perform a Design Space Exploration (DSE) to find the Pareto-optimal implementations. We model the DSE problem as a Multiobjective Optimization Problem (MOP) and we use an evolutionary approach to rapidly converge towards the Pareto-front. Moreover, we provide the design of a new majority voter adapted to the new proposed architecture. Experiments demonstrate the advantages of our approach. Indeed, results show that for 97% of the examined circuits it was possible to find QAMR variants achieving a gain – compared to the TMR counterpart – either in terms area or timing (∼75%) or even in terms of both (∼22%).

Keywords. Keywords—Combinational circuits; fault tolerance; error correction; triple modular redundancy; approximate computing.

1 Introduction

During the lifespan of a system used in harsh (e.g. radiative) environment, its hardware is subject to various physical phenomena that may alter its performance or provoke errors [ref1]. More specifically, circuits manufactured with advanced nanometer technologies are prone to errors. Effects like aging or wear-out lead to permanent fault and hence hard errors; energetic charged particles, instead, lead to transient faults and hence soft errors. When the fault effects propagate through the logic, they can be captured by a memory cell (e.g. a flip-flop) and stored as erroneous values. Ultimately, this may cause a malfunction of the system. In response to the stress experienced by the circuits during their lifespan, several fault-tolerant mechanisms have been designed in the years. A well known existing fault-tolerant architecture – capable of tolerating soft and hard errors – is the Triple Modular Redundancy (TMR). A triplication of

the circuit and a majority vote ensure an extreme logic error masking at a cost of a 200% area and power overhead. A TMR is capable to tolerate permanent or transient faults occurring in one or several modules – provided that they do not impact the same outputs.

Approximate Computing (AxC) is an emerging computation paradigm relaxing non-critical specification of a computing system to achieve gains in terms of resources. The underlying idea is that, for some applications, an inaccurate result does not catastrophically impact the final outcome but can provide disproportionate savings in terms of resources [ref2]. AxC has been applied to resilient applications, e.g. speech recognition, image encoding, etc., where an approximate result is sufficient for their purpose [ref3]. From the hardware standpoint, AxC enables the creation of Approximate Integrated Circuits (AxICs) whose output values may differ from the original circuit for a certain set of input values [ref4].

The application of AxC to TMR has led to the Approximate Triple Modular Redundancy (ATMR). In [ref5], two or even three different approximate modules were used to implement the ATMR. Other proposals of a low cost TMR based on approximate computing were presented in [ref6] and more recently in [ref3] and [ref7]. The AxC application to TMR leads to lower both silicon area and power consumption overhead. However, such advantages come at the expense of a reduced error-masking capability, which makes the ATMR not suitable in safety-critical scenarios.

To overcome the above issue, in [qamr-ets] we proposed the Quadruple Approximate Modular Redundancy (QAMR) to ensure the same fault tolerance as the TMR while still benefiting from approximation advantages. To implement the QAMR, we used four approximate circuit replicas. For the QAMR to work correctly, at least three approximate replicas must provide a correct (i.e., non-approximate) response, at a given time. In other words, the four AxICs must be must be complementarily approximated with respect to the input set. In [qamr-ets], we proposed an approximation method based on selectively removing outputs and the related fan-in internal logic to form the approximate replicas. We removed a specific output from only one replica so that the other three replicas still had it. The fundamental advantage of such method is that the voter does not need to change. We explored the feasibility of the approach by using a simple random method to remove the circuit outputs.

In this paper we propose the following novelties: (i) using another approximation approach to realize the four QAMR replicas. This approach is based on the logic falsification, used in a previous contribution [logic-falsification]; (ii) the design of a new majority voter adapted to the proposed new QAMR architecture; (iii) a thorough Design Space Exploration (DSE) to find the Pareto-optimal implementations w.r.t. silicon area and timing. To validate our approach experimentally, we resorted to publicly available combinational circuits. Experimental results show that for 97% of the examined circuits it was possible to find QAMR variants achieving a gain either in terms area or in timing ($\sim$75%) or even in terms of both ($\sim$22%).

The remainder of the paper is organized as follows. Section ?? surveys previous works on fault tolerance based on AxC. Section ?? presents the QAMR approach and the proposed novelties. Section ?? details the DSE flow and shows the experimental results. Finally, Section ?? draws the conclusions.

2 State-of-the-Art on AxC Based Fault Tolerance

TMR is a fault tolerant scheme made of three identical instances of a circuit connected to a majority voter. TMR protects against faults (permanent or transient) occurring in one or several modules – provided that they do not impact the same outputs if several modules are faulty. This fault tolerant solution requires a 200% area overhead due to the two extra circuit instances. Moreover, we must add the voter area that depends on the number of circuit outputs.

AxC is able to target different layers of computing systems, from hardware to software [ref2]. In this work, we focus on AxICs, which are the outcome of AxC application at hardware level, specifically on Integrated Circuits (ICs). Approximate hardware is a design concept in which the designer selectively relaxes non-critical specifications to improve chip area, power and/or run time. Non-critical specifications usually refer to resilient applications where the level of accuracy can be lowered without impacting the system integrity. Different strategies to approximate combinational circuits have been proposed in the literature. These strategies can be grouped into the three main approaches: • Ad-Hoc approximate circuits which usually change from case to case. The Ad-Hoc approach is necessary when the designer needs to add or remove very specific functionalities to the original circuit. For example, authors in [ref8] and [ref9] propose accuracy-configurable adder and multiplier, respectively, to allow reducing the power consumption when the application can accept some inaccuracy. Although they can be efficient, Ad-Hoc approaches usually entail big design efforts when applied to large circuits. • Automatic approximate circuit synthesis methodologies, which assist the designer in reducing the cost of a circuit while minimizing the impact on the accuracy. Authors in [ref10] present an algorithm to design general inexact circuits achieving a certain Quality of Resilience (QoR). A quality function determines if the circuit meets the QoR requirements. In [ref11], the authors developed an evolutionary technique to approximate circuits until a certain approximation level is reached. Large circuits can benefit from these synthesis methodologies. • Hardware neural accelerators to implement approximate functions. Neural Networks (NNs) offer a significant parallelism capability and can be efficiently accelerated by dedicated hardware to gain in performance/energy at the expense of accuracy. For example, in [ref12], authors propose NN-based accelerators to approximate Transcendental Functions (i.e. cos, exp, log, pow, and sin).

Several proposals exist in the literature to reduce the TMR overhead by using AxC. This approach is generally known as Approximate TMR (ATMR) [ref5]. The ATMR approach uses three AxICs instead of three full-precise replicas. In these implementations, only one AxIC can give an erroneous answer at a time. In other words, each approximate module has its own unique domain of approximation. However, producing such a low cost TMR may suffer from severe limitations in term of reliability. Let us resort to an example to illustrate the above mentioned issue. Let X be an input vector for the ATMR replicas. One of the three AxIC replicas produces a wrong response – due to the approximation – when X is provided at inputs, while the other two produce a correct response. Let us imagine that a Single Event Transient (SET) occurs in one of the replicas, thus modifying its output. If the SET occurs in the replica providing the approximate output for the input vector X, then the voter will still be able to produce the correct response, thanks to the two remaining replicas. Conversely, if the replica providing the correct output experiences an SET, there will be two incorrect responses, i.e., the approximate one and the faulty one. Thus, the voter may likely produce a wrong response. In summary, input vectors for which only two out of three replicas compute correctly are not protected against SETs. In conclusion – since designing fault tolerance circuits for safety-critical applications is a crucial task – realizing it by using AxC-based schemes entails some important challenges. Specifically, it is mandatory to know the workload of such applications. For ATMR solutions, it implies that the input vectors not protected against faults must not be critical for the application. Such design requirement can be challenging and not always achievable, even for resilient applications.

3 Proposed QAMR Scheme

The goal of our approach is to achieve the same fault-tolerance capability as the TMR, while reducing the costs. We propose to make use of AxC in a quadruple duplication scheme. The final goal is to protect the functionality against permanent and transient faults for any input

vectors, as in a conventional TMR. Such fault-tolerant scheme will be suitable for safety-critical applications even when the workload is unknown.

3.1 QAMR principle

Let us resort to Figure ?? to show the approach. Let f be a generic function, whose input domain D can be split into four sub-domains $D1, D2, D3, D4$, as shown in Figure ??. The conventional TMR approach, sketched in Figure ??, uses three identical copies of the circuit implementing f and a voting scheme. This ensures the fault tolerance when one of the three circuit replicas incurs some defective conditions. In such case, the defective copy will produce incorrect outputs, for some inputs. Thanks to the other two correct copies, the majority voter can still provide the correct output. As sketched in Figure ??, the QAMR employs four circuits suitably approximated

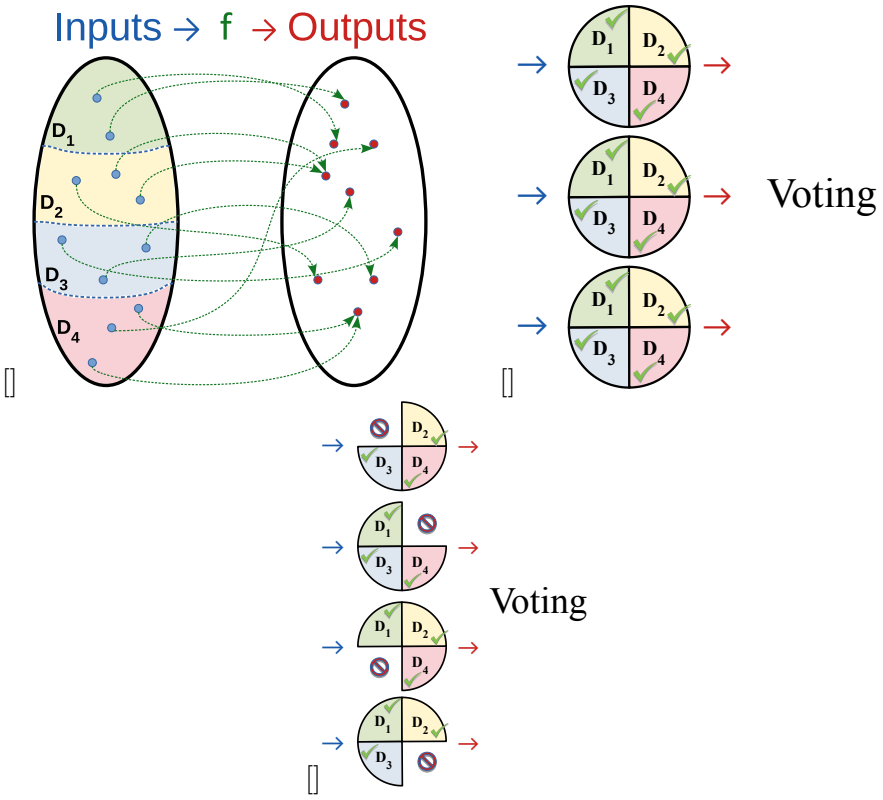


Figure 1: (a) Generic function's domain split into four (D1-D4); (b) the TMR uses three identical circuit copies: all the circuits produce the correct results for all the function's domains (D1-D4), when no errors occur; (c) QAMR uses four approximate circuit copies: each function's domain is covered three times, as in TMR.

to ensure the desired fault tolerance and achieve efficiency gains. Specifically, the four AxICs should be approximated so that all the function's domains are covered three times – as in TMR scheme. This represents the first approximation constraint to satisfy in the approximation process. At the same time, using AxIC enables the opportunity to achieve efficiency gains. The underlying insight is that a good AxC technique achieves more gains than it reduces the system accuracy.

In [qamr-ets], the approximation technique that we proposed guaranteed that the conventional majority vote could be used. This was possible because we approximated the four replicas by selectively removing outputs. By doing so, we obtained only three replicas for each output signal. In this work, the employed approximation method removes internal logic portions, while keeping all output signals. This calls for a new 4-bit voting approach. The straightforward

method is to put a selector circuitry right between the four replicas and the majority voter, in order to prevent the approximate response from propagating to the voter. There are two major drawbacks for this method: (i) the selector needs information on which replica performs the approximate computation, at any time; (ii) the selector entails a big overhead that may undermine the approximation gains. Thus, in this work we propose a new efficient voting strategy. However, this introduces a second approximation constraint. In the next subsection, we show the proposed approximation approach and in Subsection ?? we present the new voting strategy.

3.2 Approximation approach

To implement our QAMR approach, we propose a circuit approximation method to produce the AxICs satisfying the following conditions:

(i) at least three AxICs must produce a correct response for a given input vector, at the same time;

(ii) the four AxICs must produce the same approximate value for a given output.

To perform the approximation, we resort to the logic falsification, employed in a previous contribution [logic-falsification]. As logic falsification, we refer to the process of modifying the on-set or alternatively the off-set of a given Boolean function, thereby introducing errors in its definition on purpose. In Table ??, we report a simple example of approximation by logic falsification.

	Inputs			Original		$AxIC_0$		$AxIC_1$		$AxIC_2$		$AxIC_3$	
	A	B	C	out_0	out_1	out_0	out_1	out_0	out_1	out_0	out_1	out_0	out_1
$Subset_0$	0	0	0	1	0	(0)	(1)	1	0	1	0	1	0
	0	0	1	0	0	(0)	(1)	0	0	0	0	0	0
$Subset_1$	0	1	0	0	1	0	1	(0)	(1)	0	1	0	1
	0	1	1	1	1	1	1	(0)	(1)	1	1	1	1
$Subset_2$	1	0	0	0	0	0	0	0	0	(0)	(1)	0	0
	1	0	1	0	1	0	1	0	1	(0)	(1)	0	1
$Subset_3$	1	1	0	1	0	1	0	1	0	1	0	(0)	(1)
	1	1	1	1	0	1	0	1	0	1	0	(0)	(1)

Approximate output values chosen: $O_0 = 0$, $O_1 = 1$

Table 1: Example of approximation by logic falsification

We first divide inputs into four different subsets. For each subset, we apply the logic falsification (i.e., we fix the output response) to only one of the replicas. Different replicas provide the same approximate output for different input subsets. In the example, $AxIC_0$ provides out_0 = '0' and out_1 = '1' as response to the inputs in $subset_0$ and a correct response to inputs in other subsets; the same goes for $AxIC_1$ which provides out_0 = '0' and out_1 = '1' as response to the inputs in $subset_1$ and a correct response to inputs in other subsets, and so on. This satisfies both the first and the second approximation constraints. The final goal is to obtain four AxIC replicas so that the corresponding QAMR structure provides the same robustness as the conventional TMR while providing also gains in terms of resources.

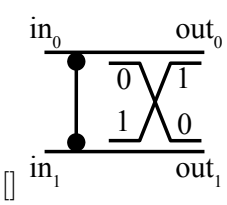
> Depending on how we choose the input subsets and the values for approximate outputs we obtain different outcomes.

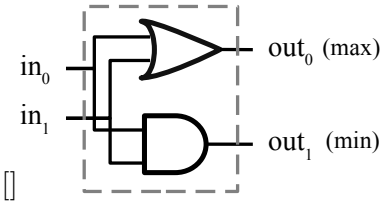
Therefore, a DSE is necessary to find the optimal choices. The next subsection shows the new voting strategy, while Subsection ?? details the DSE.

3.3 A sorting-network-based voting approach

As described in Subsection ??, we generate four AxICs, so that a given one provides approximate responses only when the other three provide non-approximate responses. To be able to correctly

and efficiently perform a majority vote, we propose a voting approach based on the well known binary Sorting Networks (SortNets) [sorting-networks]. SortNets are made of comparators and wires. A wire carries a binary value through the network. A comparator connects two wires and sorts the two values carried by the wires. In Figure ?? we report the symbolic representation of a comparator; Figure ?? represents the corresponding logic-gate implementation and Figure ?? the related truth table. As shown in the figure, the comparator employs a logic OR and a logic AND to perform the max and the min functions, respectively. The goal of the network is to

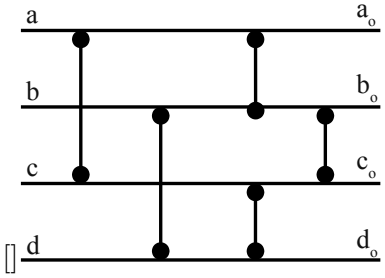




in_0	in_1	out_0	out_1
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Figure 2: (a) comparator symbol: two signals are sorted when they land on a comparator; (b) logic-gate implementation of the comparator and (c) corresponding truth table.

produce the sorted sequence of the input signals, by arranging the comparators accordingly. Our purpose is to sort the four signals produced by the four AxICs. In Figure ??, we report a four-wire sorting network representation and in Figure ?? the corresponding logic-gate implementation. This leads to clearly separate the zero ('0') values from the one ('1') values at the SortNet



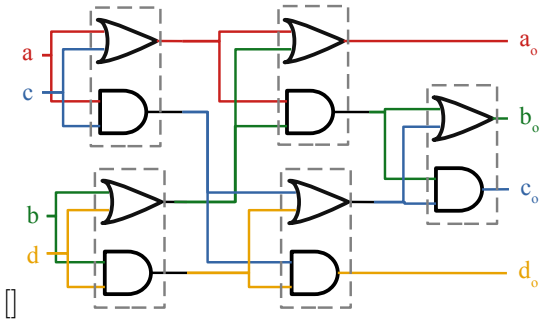


Figure 3: (a) four-wire sorting network representation and (b) corresponding logic-gate implementation.

output. Let us resort to an example to show the merit of the idea. Figure ?? depicts two simple examples where one AxIC replica produces a wrong value, due to the approximation. When the correct response for an output signal is '0' (Figure ??) and one of the replicas produces '1' due to the approximation, the SortNet arranges the values so that the wrong response ends up to the top. Conversely, when the correct response for an output signal is '1' (Figure ??) and one of the replicas produces '0' due to the approximation, the SortNet arranges the values so that the wrong response ends up to the bottom. In this way, the three non-approximate values (wrapped in the green ovals in Figure ??) can be voted. Moreover, the middle value in the oval turns out to be the majority vote output, i.e., c_o in Figure ?? and b_o in Figure ??. Indeed, as shown in the example in Figure ??, even when a fault strikes on an AxIC producing a non-approximate response and turns it into a wrong one, the SortNet-based voter is still able to deliver the correct response, i.e., $b_o = 1$.

To suitably embed in the QAMR architecture the voting structures depicted in Figure ??, we choose at design time and for each output the suitable variant of the SortNet among those in Figures ?? and ??. Indeed, thanks to the second approximation constraint, in the approximation

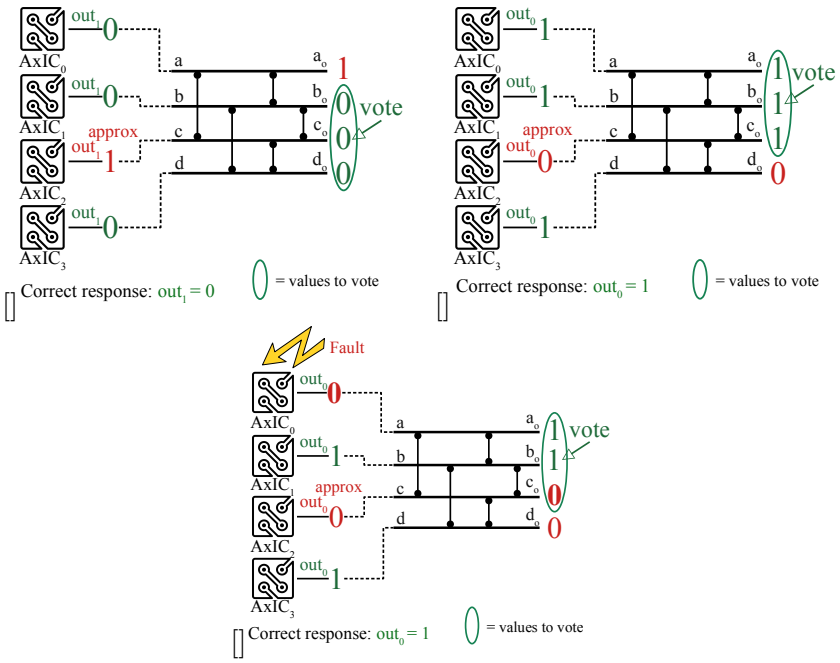


Figure 4: (a) and (b) four-wire SortNet working principle. (c) Example showing the correctness of the SortNet-based voting strategy under a fault occurrence.

process we guarantee that the AxIC replicas produce the same approximate response for a given output (see Table ??).

Finally, since we do not need all the four output values of the SortNet, we can optimize the implementation. This leads to the logic-gate implementations in Figures ?? and ??. The first

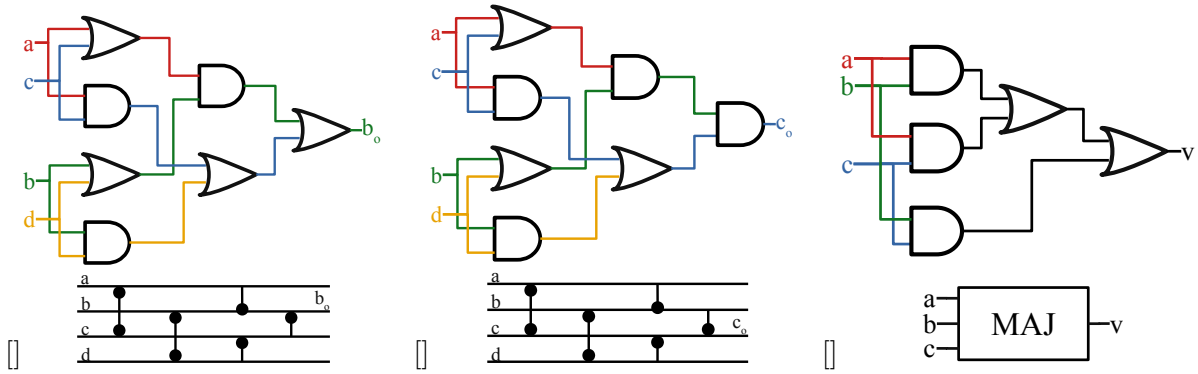


Figure 5: (a) - (b) Sorting-network-based voter implementations (see Figures ?? - ??); (c) conventional majority voter

one (??), has to be used to perform a vote for approximate outputs equal to '0' (i.e., Figure ?? and out_0 in Table ??). The second one (??), has to be used to perform a vote for approximate outputs equal to '1' (i.e., Figure ?? and out_1 in Table ??). For comparison, in Figure ??, we report the conventional majority voter logic-gate implementation. The proposed SortNet-based voter entails a moderate overhead in terms of logic gates (7 instead of 5) and still has three logic levels, thus it has no impact on the circuit response timing.

3.4 Design space exploration

We model the DSE as a Multi-objective Optimization Problem (MOP). Basically, a MOP consists of a set of fitness-functions to minimize/maximize at the same time under a set of con-

straints. Since fitness functions often represent conflicting goals, a set of equally good solutions, namely the Pareto-front, is achieved as result of the MOP resolution. Being $x, y : x \neq y$ two solutions, x is said to Pareto-dominate y i.f.f. x shows better values than y for all the considered fitness functions. If a solution is not dominated by any others, it is called a Pareto-optimal solution.

MOPs are NP-hard problems due to the rapid growth of the size of the solution space as the number of decision variables and fitness functions and constraints increases. Exact resolution algorithms turn out to be very computation-intensive. Therefore, usually they are not suitable when the search space is large. Consequently, we resort to a heuristic to produce a sub-optimal Pareto-front in a reasonable time. Genetic Algorithms (GAs) have been largely used in the literature to find Pareto-fronts for MOPs [deb2001multi, van2000multiobjective]. GAs operate on a randomly generated set of individuals, called the initial population, that evolves and, eventually, converges to a set of Pareto-dominant solutions. Each individual is represented as a chromosome, i.e. a vector in which each element, namely a gene, models and determines one or more features of the individuals. During the evolution process, a random mutation may occur: when it provides additional advantages, then new species thrive over the old ones; conversely, disadvantageous mutations lead the species to disappear in time. In this way, the selection of the best species takes place. A crossover occurs when two parent chromosomes are combined together to form a new chromosome, the offspring; essentially, crossover makes good genes appear more frequently in a population.

To model our DSE problem we chose the final silicon area of the QAMR and its delay as fitness functions to minimize. These two objectives are well known to be competing. An individual for the GA corresponds to a QAMR implementation, i.e., the four approximate replicas attributes. Thus, as chromosomes, we encoded the input subsets and the approximate output values. In this way, by mutating the chromosomes, the GA makes the QAMR implementations evolve towards a Pareto-optimal frontier. The next section describes our DSE flow in more details.

4 QAMR DSE flow and experimental results

In this section, we firstly describe the flow that we followed to perform the DSE, then we show and comment our experimental setup and results.

4.1 DSE flow

In Figure ?? we report the flow of the proposed DSE. It is based on the utilization of a GA to explore the different possible approximation opportunities and converge towards the Pareto-optimal ones. Starting from the original Boolean function, a logic optimization allows creating an optimized precise (i.e., non-approximate) function. Three replicas will form the TMR along with a conventional majority voter. We used Espresso [espresso] and ABC [abc] to perform different logic optimizations oriented to the final implementation in ASIC technology. In parallel, a first population of approximation parameters (i.e., input subsets and approximate output values) is generated randomly and expressed as chromosomes; hence, a logic falsification operation is performed, as described in Subsection ??, to obtain four approximate Boolean functions (see Table ?? for an example); a logic optimization allows creating four optimized approximate functions; these will form the QAMR along with a SortNet-based voter. Finally, the fitness function is evaluated, i.e., TMR and QAMR are compared in terms of silicon area and delay thanks to the reports from the optimization tools. The genetic algorithm keeps the most promising individuals (i.e., QAMR versions providing improvements over the TMR in terms of area and/or timing)

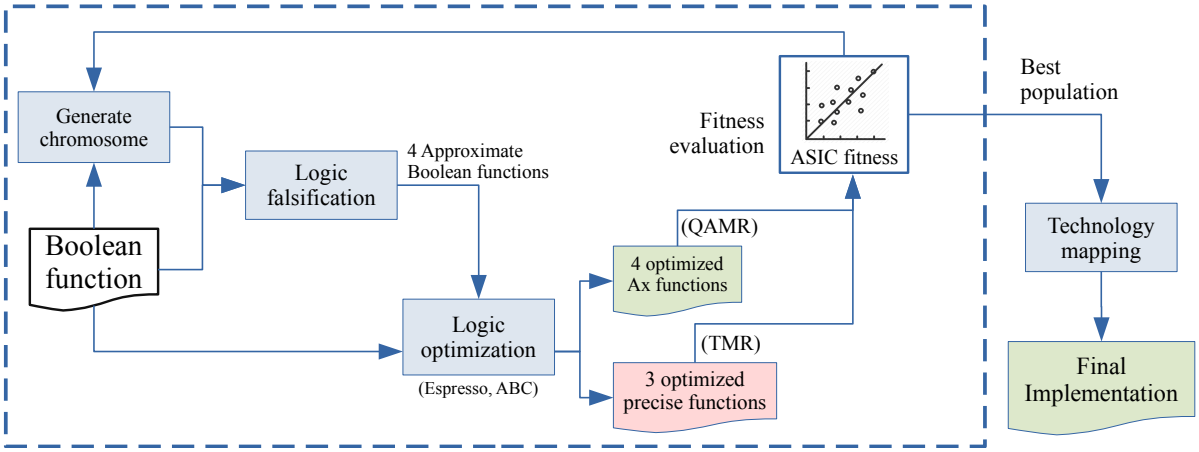


Figure 6: GA-based DSE flow.

and neglects the others. The process iterates until a final population of Pareto-optimal QAMR versions is obtained. The best population is ultimately mapped onto a technology library.

4.2 Experimental Setup

We evaluated our approach by using combinational circuits from the publicly available LGSynth'91 benchmark suite [ref13]. For each circuit, we applied the flow described in Figure ??. To map the final population we used Genus Synthesis Solution from Cadence [ref14], and the FreePDK45 45nm technology library [ref15]. Moreover, we synthesized the new proposed SortNet-based voter to compare it to the conventional majority voter.

4.3 Experimental Results

In this subsection we present experimental results to validate our approach. Firstly, we report in Table ?? the results of the synthesis for the proposed new SortNet-based voter and compare it to the conventional majority voter, in terms of silicon area and delay. We obtained the results from the Genus after-synthesis reports. As expected, silicon area overhead is moderate w.r.t.

Attribute	SortNet-based voter	Majority voter
Area	16.425499 um^2	11.7325 um^2
Delay	138 ps	138 ps

Table 2: SortNet-based voter synthesis results and comparison with the majority voter

the majority voter and the delay turns out to be the same for both voters. Considering that the voter is usually smaller than the circuits to vote for, the introduced overhead is likely to be negligible compared to the gains achieved thanks to the approximation.

As already mentioned, the result provided by the GA is a population of QAMR variants with its related Pareto-front. The results reported are expressed as relative gains, according to the following formula:

$$\text{gain} = \frac{(\text{original value} - \text{new value})}{\text{original value}} \tag{1}$$

We organize the results on a Cartesian plane where x-axis represents the silicon area gain w.r.t. TMR and the y-axis the timing gaing w.r.t. TMR. The TMR version is represented by a star in the origin of the Cartesian axes. The QAMR variants can lay in different quadrants of the Cartesian plane. Specifically: • if a variants lays in the Q1 quadrant, it achieves gains in terms of both silicon area and timing; • if a variants lays in the Q2 quadrant, it achieves gains only

in terms of timing; • if a variants lays in the Q4 quadrant, it achieves gains only in terms of area; • if a variants lays in the Q3 quadrant, it does not achieve gains, at all. As an example, in

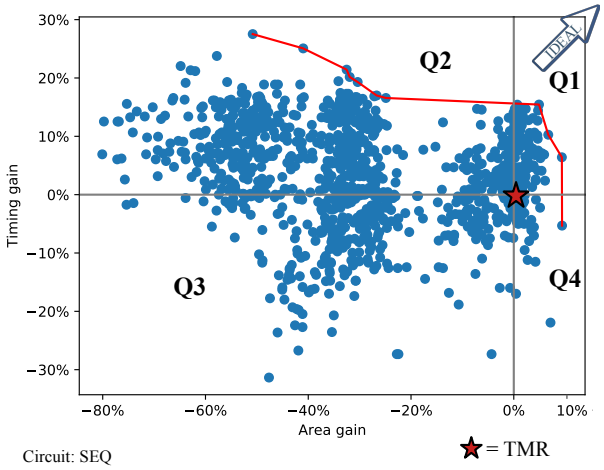


Figure 7: Example of QAMR population and relative Pareto-front generated by the GA, for the SEQ circuit

Figure ??, we report the final population for the SEQ circuit, along with the Pareto-front. The figure highlights that for a lot of QAMR variants (individuals in the population) we achieved a gain in both silicon area and timing (quadrant Q1), or only in time (quadrant Q2), or only in area (quadrant Q4); for some variants, the DSE also achieved no gains at all (quadrant Q3). More importantly, the DSE allowed finding Pareto-dominant solutions that always achieve a gain in terms of both area and timing (Q1) or at least in terms of one of them (Q2,Q4).

Finally, in Figure ?? we report the Pareto-fronts for the examined circuits from LGSynth'91 benchmark suite [ref13]. The results clearly show the advantage of the proposed approach. Indeed, for 97% of the examined circuit the DSE led to QAMR variants achieving a gain over the TMR. In details, over 41 examined circuits, 31 circuits ($\sim$ 75%) have at least one variant gaining in timing – i.e., at least one variant in Q2 – and 9 circuits ($\sim$ 22%) have at least one variant gaining in both area and timing – i.e., at least one variant in Q1. The maximum achieved gains were 30% in timing and 9% in area. Only for a single circuit (tcon) the DSE did not find any approximation leading to a gain, i.e. all the variants lay in Q3.

5 Conclusion

In the context of error-tolerant applications, approximate computing trades off some computing accuracy for efficiency gains. In this context, studies in the literature proposed to relax reliability constraints to achieve those gains. Despite the efficiency optimization opportunities brought by this kind of techniques, reliability still represents a key requirement in most advanced safety-critical computing systems: sacrificing reliability could result in the production of more cost-efficient systems, but also in endangering human lives. In particular, previous works on approximation-based TMRs presented the advantage of reducing its area cost compared to the standard TMR. However, such advantage comes at the expense of a reduced fault tolerance, preventing the Approximate TMR to be used in safety-critical applications. In a previous paper [qamr-ets], we presented the Quadruple Approximate Modular Redundancy (QAMR), a solution to profit from the benefits brought by AxC, without sacrificing the reliability requirements. The QAMR reduces the standard TMR area cost without sacrificing fault tolerance capabilities. The previous work was based on approximating the circuits by removing some output cones. In this work, we proposed to use another approximation approach, i.e., logic

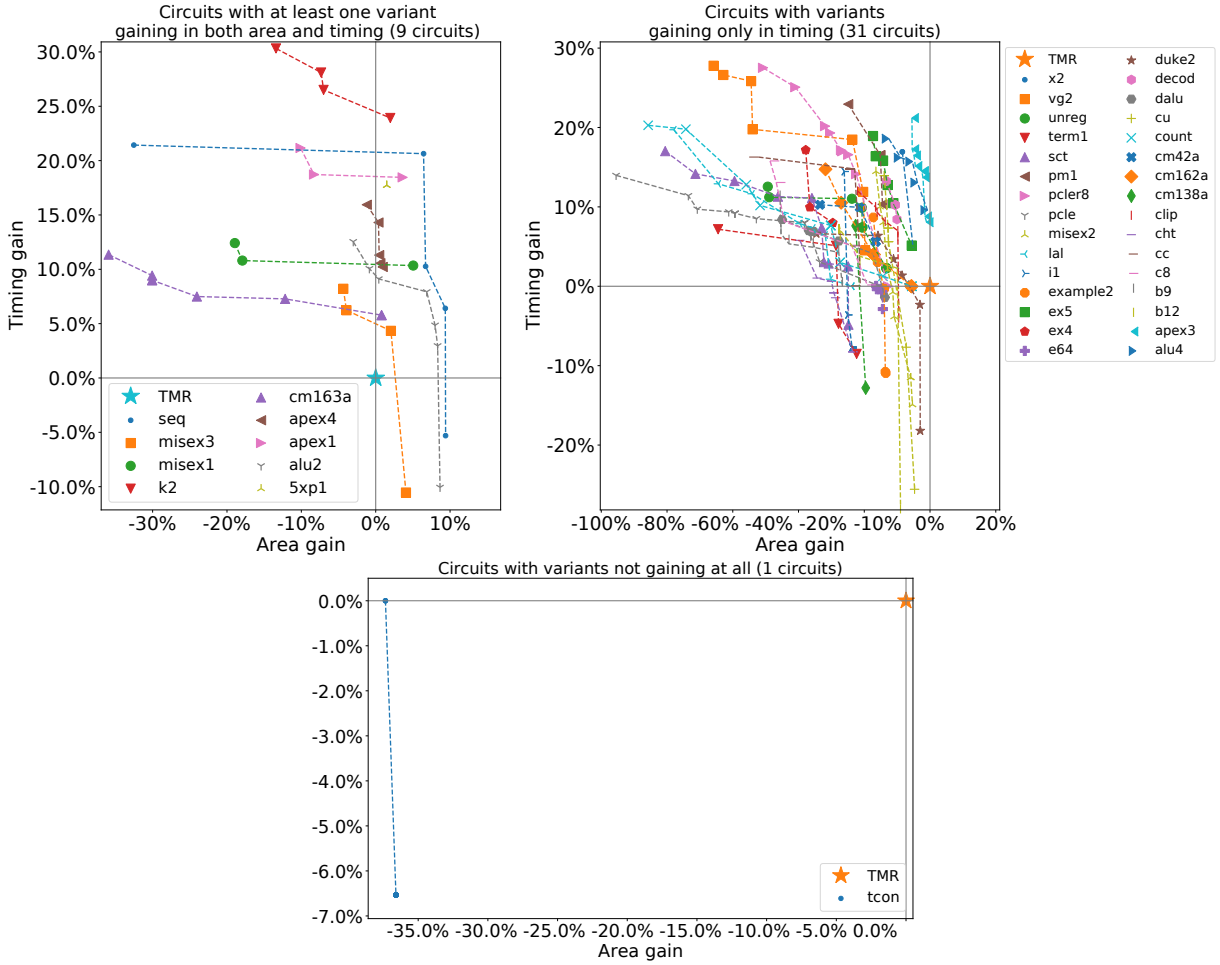


Figure 8: Pareto-fronts resulting from the DSE. The results are expressed in terms of area gain and timing gain.

falsification, to explore further opportunities for the QAMR architecture. Moreover, we provided the design of a new sorting-network-based majority voter adapted to the new proposed architecture. Finally, we performed an extended DSE to find the Pareto-optimal QAMR configurations. Results show that for 97% of the examined circuits it was possible to find QAMR variants achieving a gain either in terms area or in timing (∼75%) or even in terms of both (∼22%). DSE with a different target technology, such as FPGA, will be performed as a future work, as well as a thorough comparison with previous work.