DOI 10.1109/TR.2016.2604918

Error Mitigation using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches

Antonio J. Sanchez-Clemente, Luis Entrena, Radek Hrbacek, Lukas Sekanina

***Abstract*—Technology scaling poses an increasing challenge to the reliability of digital circuits. Hardware redundancy solutions, such as Triple Modular Redundancy, produce very high area overhead, so partial redundancy is often used to reduce the overheads. Approximate logic circuits provide a general framework for optimized mitigation of errors arising from a broad class of failure mechanisms, including transient, intermittent and permanent failures. However, generating an optimal redundant logic circuit that is able to mask the faults with the highest probability while minimizing the area overheads is a challenging problem. In this work we propose and compare two new approaches to generate approximate logic circuits to be used in a TMR schema. The probabilistic approach approximates a circuit in a greedy manner based on a probabilistic estimation of the error. The evolutionary approach can provide radically different solutions that are hard to reach by other methods. By combining these two approaches, the solution space can be explored in depth. Experimental results demonstrate that the evolutionary approach can produce better solutions, but the probabilistic approach is close. On the other hand, these approaches provide much better scalability than other existing partial redundancy techniques.**

***Index Terms*—Approximate logic circuit, error mitigation, evolutionary computing, Single-Event Transient, Single-Event Upset.**

Acronyms and Abbreviations

CGP	Cartesian Genetic Programming
DWC	Duplication With Comparison
EDAC	Error Detection And Correction
MA	Mandatory Assignment
SAT	Satisfiability
SMA	Set of Mandatory Assignments
SET	Single-Event Transient
SEU	Single-Event Upset
TMR	Triple Modular Redundancy

Notation

EP	Total Error Probability
EA	Estimated Area
$P(f)$	Probability of testing a fault f
G	Original circuit with no approximation
H	Over-approximate circuit
F	Under-approximate circuit

This work was supported by the Ministry of Economy and Competitiveness of Spain under project ESP2015-68245-C4-1-P, and by the Czech science foundation project GA16-17538S and the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science - LQ1602.

I. Introduction

The impact of transient, intermittent and permanent failures on digital circuits is steadily increasing with technology scaling. Circuits manufactured on advanced technologies are more prone to errors due to several reasons, which are primarily related to the shrinking of transistor dimensions and the increase in the total number of gates per chip [1].

On the one hand, manufacturing process variations are a dominant source of static variability which may significantly affect yield. As a consequence, it is now common practice to use error correction codes or hardware redundancy in circuits with a regular internal structure, such as memories or programmable logic devices (FPGAs). In the past, variations were mostly due to imperfect process control, but now intrinsic atomistic effects, such as Random Dopant Fluctuations (RDF) or Line Edge Roughness (LER) have become relevant in sub-45-nm technologies, as devices of atomic sizes are achieved [2]. Due to the increasing difficulty of testing, some defects may escape manufacturing test and may cause intermittent failures resulting in errors during normal operation. Furthermore, transistor aging effects, such as negative-bias temperature instability (NBTI), also increase intermittent gate failures during the lifetime of a chip. Manufacturing variations, supply voltage variations, temperature variations and aging-related effects in digital circuits pose an increasing challenge to reliability [3].

Radiation-induced soft errors caused by ionizing particles, mainly neutrons at the atmospheric level and other particles in space environments have also become a big concern. In the past, transient effects in memory elements, known as Single-Event Upsets (SEUs), were the primary concern. However, for advanced technologies SEU protection is not enough, as transient effects in combinational logic gates, known as Single-Event Transients (SETs), are becoming very relevant [4]. Protection against SETs is much more difficult to achieve and typically involves a large amount of redundancy. Finally, post-silicon technologies such as carbon nanotubes are intrinsically less robust and require fault-tolerance [5].

Hardware redundancy is often used in safety- and mission-critical applications to mitigate the effects of transient errors, permanent errors or configuration errors in FPGAs. Duplication With Comparison (DWC) or Triple Modular Redundancy (TMR) are well-known examples of techniques that provide concurrent error detection and correction capabilities, respectively. However, these techniques typically introduce very large overheads, which is more than 200 % in the case of TMR. When such overhead is not acceptable, partial redundancy is used in order to find a good balance between the reliability requirements and the area, power and performance requirements [6].

Approximate logic circuits provide a general framework for optimized mitigation of errors arising from a broad class of failure mechanisms, including transient, intermittent and permanent failures. An approximate logic circuit is a circuit that performs a possibly different but closely related logic function to the original circuit. As it is not required to exactly match the original circuit, the approximate circuit can be smaller but it can still be used to detect or correct errors where it overlaps with the original circuit. Approximate logic circuits can be used in TMR instead of exact copies of the original design and the designer can select the level of approximation. A closer approximation provides higher fault tolerance but also increases the area and power. In contrast, this continuous trade-off is not possible when exact TMR is used. However, generating an optimal redundant logic circuit that is able to mask the faults with the highest probability while minimizing the area and power overheads is a challenging problem.

In this work we propose and compare two new approaches to generate approximate logic circuits to be used in TMR. First of all, a probabilistic approach is proposed which is based on dynamic probability estimations. This approach takes advantage of strongly coupling the approximation method and the error estimation method by using stuck-at faults. Departing from the original target circuit, approximate logic circuits are built by iteratively forcing original circuit lines to constant values. The reduction in error mitigation that is produced by this type of transformations can be related to the probability of detecting associated stuck-at faults, which is the metric commonly used to estimate the error coverage. This approach can be used in a greedy manner to remove the logic with the lowest probability of producing an error while the required reliability target is met. However, it is well-known that greedy algorithms may often produce suboptimal results because they may get stuck in local minima. The second approach we propose in this work is based on evolutionary algorithms. Evolutionary algorithms are used in many applications to solve hard optimization and design problems. As the method is intrinsically based on the trial and error approach, it is usually very time consuming, but, on the other hand, capable of discovering solutions hard to reach by other methods. One of the major advantages of evolutionary algorithms is the ability to get out from local minima and increase the chances to reach global minima. Thus, evolutionary algorithms can provide radically different solutions. A comparison with the probabilistic approach is carried out in this work in order to contrast their respective capabilities.

This paper is organized as follows. Section II summarizes previous work and introduces approximate logic circuits and evolutionary circuit design. Section III describes the probabilistic approach. Section IV deals with the evolutionary approach. Experimental results are presented in Section V. Conclusions are given in Section VI.

II. Previous work

Fault-tolerance techniques are classically classified into hardware redundancy, information redundancy and time redundancy techniques [7]. Among the hardware redundancy techniques, Triple Modular Redundancy (TMR) is a well-known error masking technique that is widely used in critical applications. TMR can be used at different levels of abstraction, from system to transistor level, and can protect against transient and permanent errors. Information redundancy techniques, such as Error Detection and Correction (EDAC) codes, can be very effective for single or double errors. Thus, EDAC codes are typically applied to memories or communication protocols, but they cannot be used in the general case because a low multiplicity of errors cannot be guaranteed. Finally, time redundancy is intrinsically non-robust to permanent failures and may introduce severe performance penalties.

The capability of TMR to mitigate both transient and permanent errors makes it a good technique to tackle the variety of potential failure mechanisms that must be considered for advanced technologies. However, TMR suffers from high overhead in terms of area and power (more than 200 %). To alleviate this overhead, alternative techniques have been proposed based on partial error masking. Without loss of generality, we will focus on combinational circuits. The extension to sequential circuits is trivial by applying TMR to the sequential elements along with the combinational elements.

An early partial error masking approach is proposed in [8] which consists on triplicating and voting the nodes with the highest soft error susceptibility. Subsequent approaches attempt to insert redundancies that protect against the most common errors or to resynthesize the circuit to improve reliability. In [9], an approach is proposed to provide protection for the most common output combinations. In [10], the authors propose the use of implications to build redundant logic that checks for violation of behavioural constraints.

In a recent work [11], small sub-circuits have been extracted and resynthesized using two-level techniques and fast extraction algorithm. The resynthesized circuits have been then merged to produce the final fault-tolerant circuit. Combinational restructuring has been used in [12] to improve the masking properties of a circuit. This approach takes advantage of conditions already present in the circuit, such as observability don't-cares. Other approaches use wire addition and removal for combinational restructuring [13]. Finally, there are approaches that use approximate logic circuits [14] for partial error detection and masking. These approaches are reviewed in the following section.

A. Approximate logic circuits

The concept of approximate logic circuit or function provides a systematic framework for the implementation of fault-tolerant combinational logic circuits. Given a logic function G, an approximate logic function is a function $\hat{G}$ that performs a possibly different but closely related function. The approximation divides the input space into two subspaces: the subset of input vectors for which G and $\hat{G}$ produce the same output (correct subspace) and the subset of input vectors for which

G and $\hat{G}$ produce different outputs (incorrect subspace). The quality of an approximation is evaluated as the relative size of the correct subspace.

Approximations can be classified as unidirectional or bidirectional [15]. An approximation is called unidirectional if the incorrect subspace is either a subset of the on-set or a subset of the off-set of G. In the first case, $\hat{G}$ is called an under-approximation or on-set unidirectional approximation of G. Similarly, in the second case $\hat{G}$ is called an over-approximation or off-set unidirectional approximation of G. In the sequel, the under-approximations and over-approximations of a function G will be denoted respectively as F and H.

By definition, a unidirectional approximation satisfies an implication relationship. If F is an under-approximation of G, then $F = 1 \Rightarrow G = 1$ and, conversely, $G = 0 \Rightarrow F = 0$. The incorrect subspace corresponds to the input vectors that produce $G = 1$ and $F = 0$, i.e., all input vectors in the incorrect subspace produce unidirectional $1 \rightarrow 0$ errors. If H is an over-approximation of G, then $H = 0 \Rightarrow G = 0$ and, conversely, $G = 1 \Rightarrow H = 1$. In this case, the incorrect subspace corresponds to the input vectors that produce $G = 0$ and $H = 1$, i.e., all input vectors in the incorrect subspace produce unidirectional $0 \rightarrow 1$ errors. Bidirectional approximations do not satisfy an implication relationship and can produce both $0 \rightarrow 1$ and $1 \rightarrow 0$ errors.

Partial logic masking can be obtained by using a TMR schema in which two of the copies are replaced by approximate logic circuits, as shown in Fig. 1. Note that the approximate logic circuits may produce incorrect outputs even in the absence of faults. To ensure these incorrect outputs are masked, it is required that the incorrect subspaces of the two approximations do not overlap, so that at most one of the circuits is allowed to produce an incorrect output for any input vector. This condition is met by using an under-approximation F and an over-approximation H in the TMR schema.

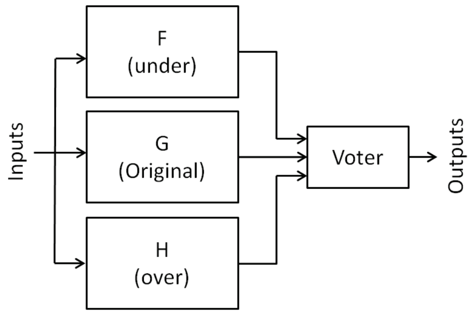

Fig. 1. Error masking schema using approximate logic circuits

The error masking capabilities of this schema can be better explained using the diagram shown in Fig. 2 [14], where the on-sets of the original and the approximate logic functions are represented. In the areas where the original and the approximate logic functions overlap (correct subspace), all circuits produce the same output value. Because the three circuits are implemented separately, a single fault can only affect one of them at a time and its effect will be masked. In the areas where the approximate functions do not overlap (incorrect subspace), one of the approximate functions produces an incorrect result. This incorrect result is masked, but a fault in any of the other two circuits may cause an additional incorrect result which cannot be masked by the majority voter. Therefore, the probability of error is directly related to the probability of faults that can propagate errors to the outputs for input vectors in the incorrect subspace of any of the approximate circuits. The goal is to find approximate circuits that minimize this probability and can be implemented with a reduced amount of logic.

It must be noted that approximate logic circuits are susceptible to errors that may not be masked. This may happen in the incorrect subspace, if a fault in an approximate circuit causes the two approximate circuits to agree on an incorrect result. However, this situation is detectable, because it is impossible by construction that the two approximate circuits disagree with the original circuit unless there is a fault. Thus, all errors produced in the approximate logic circuits are either masked or detectable. Generally, the contribution of the approximate logic circuits to the error rate is compensated by the error masking on the original circuit. However, if this is not the case, the voter can be complemented with an error detector. This way, it is guaranteed that the failure probability always reduces as the quality of the approximation increases.

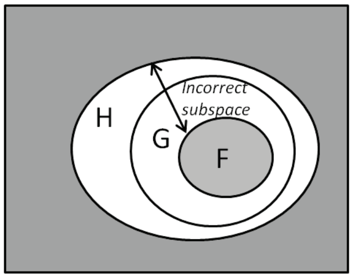

Fig. 2. Graphical representation of the relationship among the original and the approximate functions

An algorithm for technology-independent synthesis of approximate logic functions is proposed in [16]. This algorithm utilizes technology-independent networks and tries to approximate the local logic function of a node by moving minterms from its off-set or on-set into don't-cares. Minterm selection is based on the logic function of the node or, alternatively, local observability don't-cares can be used to expand the space from which minterms can be selected. However, as this approach may lead to an incorrect approximation, a SAT solver is used to ensure correctness. This approach is extended in [15] by considering predictor-indicator bidirectional approximations. This type of approximations use a predictor function, that predicts the value of the function, and an indicator function, that indicates uncertainty about the predicted value. The advantage of this approach is that the predictor and indicator functions are not required to have implication relationships with the original function G. However, predictor-indicator bidirectional approximations cannot be used when the bidirectional approximate circuit is vulnerable to errors [15].

Other implication-based approximation methods are proposed in [14], [17]–[20]. The approach in [18] considers the failure probabilities of the gates and uses a two-level representation. Finally, [19] approximates a circuit by removing circuit lines with low testability. However, this method does not allow to estimate the error probability produced by the approximation transformations. An improved probability estimation method is proposed in [20], but it does not take into account the possible faults that may occur in the approximate circuits. The probabilistic approach proposed in this work extends these techniques by considering a dynamic probability analysis and considering all faults that may occur in the original and the approximate circuits to estimate the total error probability.

B. Evolutionary Circuit Design

Since the very beginning of the research in evolutionary computation, evolutionary algorithms have been applied for purposes of hardware optimization. Several monographs [21], [22] summarize the applications from the field of electronic circuits design, diagnostics, and testing. Later, evolutionary algorithms were applied to generate complete circuit structures (i.e., not only to optimize parameters of existing circuits) and dynamically adapt circuit structures [23]. For example, in the area of dependability, an evolutionary repair method was proposed for TMR implemented into FPGAs [24]. It employs an evolutionary algorithm to repair one damaged module of TMR by using the two healthy modules as sources of golden data for the fitness function. An analysis has shown a significant improvement of reliability for small benchmark circuits.

The evolutionary design of combinational circuits has been well established in the past. Majority of designs in this area is conducted by Cartesian genetic programming (CGP) or methods similar to CGP. CGP is a branch of genetic programming (GP) introduced by Miller and Thomson [25]. Unlike GP, which uses tree representation, an individual in CGP is represented by a directed acyclic graph of a fixed size. The candidate circuits can have multiple outputs and intermediate results can be reused (see details in Sect. IV). CGP can be used to design various types of circuits as surveyed in [26].

The trickiest component of the evolutionary circuit design is formulating the fitness function. It usually contains several objectives (functionality, area, delay etc.) where the functionality is typically understood as the quality of the candidate circuit measured as the number of correct output bits compared to a specified truth table (i.e. the Hamming distance). In order to obtain a fully working circuit, all combinations of input values have to be evaluated. For a circuit with n_i inputs and n_o outputs, 2^{n_i} test vectors need to be fetched to the primary inputs and $n_\mathrm{o} \cdot 2^{n_\mathrm{i}}$ output bits have to be verified so as to compute the fitness value. The fitness calculation is computationally very intensive, since the number of test vectors grows exponentially with the number of primary inputs. Recently, it has been sped up by applying parallelism at various levels (data, thread, process) [27] or by introducing formal methods based on, for example, SAT solving [28].

When designing digital circuits with respect to multiple secondary objectives, e.g. area, latency, power consumption, or with the goal to approximate circuit behavior, one can make use of several approaches. The single-objective approach can be extended to deal with multiple objectives either by combining the objectives in a single fitness function just by summing the particular fitnesses weighted with a constant or, in a more sophisticated way, by introducing a multi-stage fitness function activating the particular objectives step by step. Thanks to the fixed size of the CGP genotype, resources can be constrained in order to find circuits with smaller area or power consumption [29]. Recently, a truly multi-objective approach to the design of (approximate) digital circuits has been proposed [30]. None of these methods, however, has been used to approximate TMR circuits.

III. Probabilistic Generation of Approximate Logic Circuits

In our proposed approach, approximate logic circuits are obtained from the original circuit by iteratively performing some logic transformations. Note that these transformations are not required to preserve the original logic functionality, but rather to simplify the logic at the expense of deviating from the original behaviour and hence reduce the error coverage. Thus, the quality of an approximation transformation is characterized by two major parameters: the error probability increment and the area savings. Previous works mostly focus on the latter and use synthesis techniques to simplify the logic. However, the impact of approximations on the final error probability can hardly be estimated during the synthesis process and hence these methods offer limited scalability.

In our approach, the error probability and the synthesis transformations are linked through the stuck-at fault concept. The stuck-at fault model is commonly used to model permanent faults. Stuck-at fault simulation is also a common approach to estimate the error rate [11], [16]. For each fault, the error probability is estimated as the fraction of input vectors that test the fault. For a set f_i of N possible faults, the total error probability EP can be computed as the average probability of testing every possible fault:

$$EP = \frac{\sum P(f_i)}{N}$$

This average can be weighted by the probability of each fault occurrence, if such information can be estimated. In the case of SETs, we can use derating factors to take into account timing and electrical masking.

On the other hand, the stuck-at fault concept is in the foundation of a class of powerful logic synthesis techniques [31], [32]. If a line stuck-at fault cannot be tested by any input vector, then the line is redundant and can be removed. Otherwise, if a line stuck-at fault is testable, the removal of the line creates a discrepancy with the original circuit. In exact synthesis, such discrepancy must be removed by adding some logic elsewhere [32], [33]. However, in approximate synthesis, the discrepancy is allowed. A preliminary approach that exploits this technique has been proposed in [19].

A. Line approximation

If a stuck-at fault in a line has low testability, it means that there are few input vectors that can test the fault. Then, a good approximate circuit can be built by assigning the line to a constant value. We refer to this transformation as line approximation. The error probability associated to this transformation is proportional to the probability of the input vectors which test the stuck-at fault. On the other hand, the area savings can be estimated as the logic that is removed by assigning the line, including the logic previously used to drive the line and the logic that can be simplified by propagating the constant value from the line.

The error produced by a line approximation is unidirectional if all the propagation paths from the line to the primary outputs have either an even or odd number of inversions. A line that meets this condition is said to have parity. The approximation of a line with no parity does not produce an under-approximation or an over-approximation. However, it is possible to approximate a circuit in lines with no parity by applying a simple transformation to the circuit. All lines in a circuit can be forced to have parity, except for possibly the primary inputs, by duplicating the gates and splitting the lines with no parity into two subsets with even and odd parities respectively [34]. This temporarily creates a larger circuit that allows the application of line approximations. Duplications that are not removed after approximation can be removed later on by resynthesizing the approximate logic circuit.

If a line l with even parity is approximated by assigning it to a logic 0, then the incorrect subspace is made up of the input vectors that test the fault l stuck-at 0. For these vectors, a $1 \rightarrow 0$ error is propagated without inversion to at least one output. Thus, the result is an under-approximation. If the line is assigned to a logic 1, then the incorrect subspace is made up of the input vectors that test the fault l stuck-at 1. In this case a $0 \rightarrow 1$ error may be propagated without inversion and the result is an over-approximation. If the line l has odd parity, the under-approximation and over-approximation are obtained with the opposite logic assignments. In conclusion, the two stuck-at faults of each line in a unidirectional circuit can be associated to either the under-approximation or the over-approximation, respectively.

Fig. 3 shows a logic circuit example and its K-map. Two approximations are shown on the right of the figure. The first one is obtained by making $d = 1$ and the incorrect subspace is highlighted in the resulting K-map. The result is an over-approximation because all errors are $0 \rightarrow 1$ errors. The second one is obtained by making $g_1 = 0$. This is an under-approximation because all errors are $1 \rightarrow 0$ errors.

Approximate circuits can be built in this way by selecting a subset of lines for approximation. In [19], all lines whose fault probability is below a selected threshold are approximated. In this work we follow an iterative approach. In each iteration, the least testable fault is selected for approximation and the effect of the line approximation on the error probability EP and the area are estimated. Iterations continue until the error probability or the area estimation reach the required target. The approximation space can be traversed with fine resolution, because each single line approximation produces a small impact on both the error probability and the area.

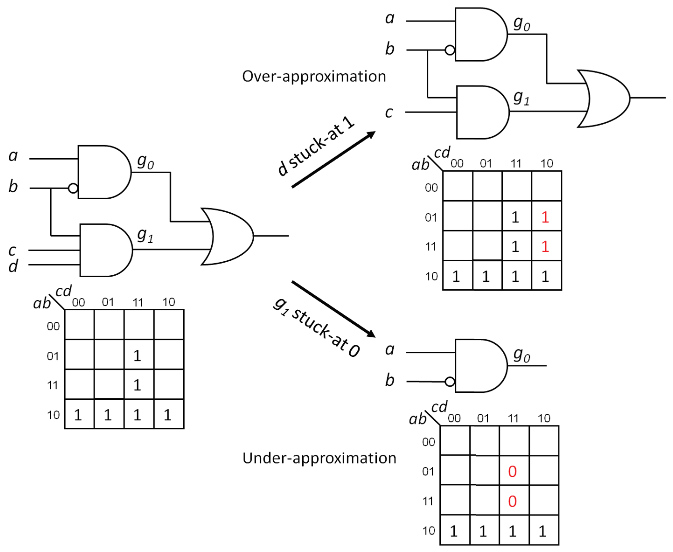

Fig. 3. Circuit approximation example

B. Dynamic Probability Analysis

It must be noted that after performing an approximation by making a line of the circuit constant, the testability of the remaining faults change. This can be illustrated with a very simple example.

Consider a 4-input AND gate, as shown in Fig. 4, and assume all of the 16 input vectors are equally likely. For a stuck-at 1 fault at any of the inputs, there is only one input vector that can test the fault, so that the detection probability is 1/16. We can then approximate the circuit by forcing one of the lines to a constant 1. The resulting approximate circuit is a 3-input AND gate. If we want to approximate additional inputs, we must take into account that the detection probability of the stuck-at 1 faults at the remaining inputs is no longer 1/16 but 1/8. Subsequent approximations at the inputs will further increase the probability to 1/4, 1/2 and 1 (when all the inputs are removed). Thus, the initial error probability estimation is less and less accurate as more approximations are taken. This requires a dynamic probability update every time an approximation is taken.

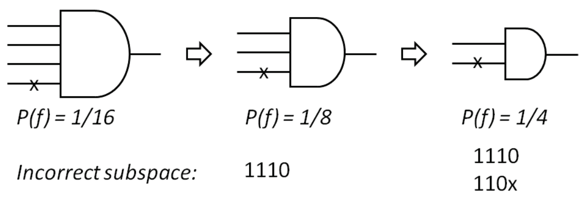

Fig. 4. Dynamic probability with successive approximations.

The problem of estimating the probability of testing a fault is generally related to testability analysis. The goal of testability analysis is to measure the difficulty of testing a fault. This measure is typically used to identify hard-to-test faults and to predict the quality of random test pattern generation. Many algorithms for testability analysis have been proposed [35]–[38] which mainly use probabilistic approaches. As the estimation of probabilities has to be updated for every circuit approximation, we need a fast and incremental algorithm. The approach used in this work was originally proposed in [20]. We summarize it here for the sake of completeness.

The notions of signal probability were established by Parker and McCluskey in [39] and are well-known in the literature. The probability that a stuck-at fault is detected by a random input vector, also called detectability, can be formulated in terms of signal probabilities [40]. The detection of a fault requires an input vector that is able to set a control value at the fault site (controllability) and to sensitize at least one propagation path (observability). These conditions can be expressed as Mandatory Assignments (MAs), so that the probability of testing a fault f can be computed as the joint probability of these MAs [41].

For instance, let us consider again the example in Fig. 3 and the fault d stuck-at 1. For this fault, the controllability assignment is $d = 0$ and the observability assignments are $b = 1$, $c = 1$ and $g_0 = 0$. Thus, the probability of testing this fault can be expressed as

$$P(f) = P(\mathrm{SMA}_f) = P(b = 1, c = 1, d = 0, g_0 = 0)$$

where SMA_f is the Set of Mandatory Assignments for the fault f. From here on we will use indistinctly $P(f)$ or $P(\mathrm{SMA}_f)$ to denote the probability of testing a fault f.

Signal probabilities in a combinational network can be easily computed by traversing the circuit from inputs to outputs [39]. At each node, the signal probability of the output is obtained as a function of the probability of the input signals according to the node type. Fault detectability can be computed as the product of the probabilities of all MAs [42]. However, this approach is only correct if all MAs are independent. To improve the accuracy of probability estimations, we use implication reasoning. Implications are actually a consequence of the existence of signal dependencies, so that signal dependencies can be removed by implying the MAs backward and forward. Then, the fault detection probability is computed as the joing probability of the final set of backward implications which cannot be further justified. If the implication of the SMA leads to an inconsistency, then we can conclude that the fault is redundant and its probability is 0. The experimental results shown in [20] demonstrate that this approach produces good estimations in comparison with stuck-at fault simulation.

In the example of Fig. 3, let us consider now the fault g_1 stuck-at 0. In this case we have $\mathrm{SMA}_f = \{g_0 = 0, g_1 = 1\}$. If all inputs are equally likely, then $P(g_0 = 0) = 0.75$, $P(g_1 = 1) = 0.125$ and $P(f) = P(\mathrm{SMA}_f) = 0.75 \cdot 0.125 = 0.09375$. This result is not accurate, but we can use implications to obtain the correct probability. In particular, $g_1 = 1$ implies $b = c = d = 1$ and $b = 1$ justifies $g_0 = 0$. Therefore, $\mathrm{SMA}_f = \{b = 1, c = 1, d = 1\}$. The product of the probabilities of these MAs gives the correct result $P(f) = 0.125$.

C. Probability-based approximation

To generate approximate logic circuits, we depart from a TMR circuit using three exact copies of the original circuit. In this circuit, when no approximation has been taken yet, any error is masked by the voter and $EP = 0$.

Consider the circuit in Fig. 1 which consists of the original circuit G, an under-approximation F and an over-approximation H. F and H have been obtained from G by approximating some lines. Let A_F and A_H be the set of faults that have been approximated to obtain F and H, respectively. The incorrect subspace is the set of input vectors that test a fault in A_F or A_H. Note that the test vectors for the faults in A_F and A_H do not overlap. Because of the approximations, some input vectors can produce a $0 \to 1$ error in F and some others can produce a $1 \to 0$ error in H, but it is guaranteed by construction that at least one of the two approximate circuits produces the correct value. Therefore, no error is observed in the absence of faults even though one of the approximations may produce a wrong value for some input vectors.

When a fault occurs in one of the three circuits, it may produce an error. In the correct subspace, this error is masked because the other two circuits are correct. However, in the incorrect subspace, one of the two approximate circuits, F or H, is also producing an error. Therefore, two of the three circuits are wrong and the error is unmasked. More precisely, errors may happen in the following three cases:

1) A fault f_G in the original circuit G may produce an error only if it propagates to the output for an input vector that tests a fault in A_F or A_H. The error probability in this case is $P(f_G \cap (A_F \cup A_H))$. Because A_F and A_H are disjoint, this probability can be computed as the sum of two terms, $P(f_G \cap A_F)$ + $P(f_G \cap A_H)$.
2) A fault f_F in F that produces an error $0 \to 1$ in F for an input vector that tests a fault in A_H. The error probability in this case is $P(f_F \cap A_H)$.
3) A fault f_H in H that produces an error $1 \to 0$ in H for an input vector that tests a fault in A_F. The error probability in this case is $P(f_H \cap A_F)$.

Note that faults that occur in an approximate circuit, F or H, may contribute to an unmasked error only in one direction. It is guaranteed by construction that faults that occur in the opposite direction either correct the error created by the approximation or cannot propagate in the incorrect subspace. On the other hand, the number of faults in F and H becomes smaller as more approximations are performed. As a consequence, the contribution of the last two cases is typically smaller than the first one. Notwithstanding, we consider all three cases in our algorithm.

To compute the probabilities in each case, we keep the SMA of each approximated line. Then, the probability of the incorrect subspaces given by A_F and A_H can be computed as the probability of the union of these SMAs. When a new approximation is performed, the new SMA is added to the set.

Fig. 5 shows the pseudo-code of the approximation algorithm. In the initialization step, we create F and H which are exact copies of the original circuit, so the total error probability EP is 0 and the estimated area EA is three times that of

```
Inputs: Original circuit (G), Estimated Area (EA),
        Error Probability Target (EP_T), Area Target (EA_T)
Outputs: Under-approximate circuit (F), Over-approximate circuit (H),
         Estimated Error Probability (EP), Estimated Final Area (EA)
// Initialization
$F = G, AF = \phi, EA_F = EA$
$H = G, AH = \phi, EA_G = EA$
$EP = 0$
$EA = EA_F + EA_G + EA_H$
// Compute initial fault probabilities
foreach fault $f_i$ in $G$
    Imply $SMA(f_i)$
    Compute $P(f_i)$
    Copy $SMA(f_i)$ and $P(f_i)$ to equivalent faults in $F$ and $H$
// Approximation loop
while ($EP < EP\_T$ or $EA > EA\_T$){
    select fault $f_A$ with min probability;
    if $f_A$ is an under-approximation fault then
        $A_F = A_F \cup f_A$
        foreach fault $f_G$ in G
            $P(f_G) = P(f_G \cap A_F)$
        foreach fault $f_H$ in $H$ that propagates as $1 \rightarrow 0$
            $P(f_H) = P(f_H \cap A_F)$
        Update $EP$
        Approximate $f_A$ in $F$
        foreach fault $f_F$ in $F$
            Update $SMA(f_F)$
            if $f_F$ is redundant, then approximate $f_F$ in $F$
        Update $EA$
    else // $f_A$ is an over-approximation fault
        $A_H = A_H \cup f_A$
        foreach fault $f_G$ in G
            $P(f_G) = P(f_G \cap A_H)$
        foreach fault $f_F$ in $F$ that propagates as $0 \rightarrow 1$
            $P(f_F) = P(f_F \cap A_H)$
        Update $EP$
        Approximate $f_A$ in $H$
        foreach fault $f_H$ in $H$
            Update $SMA(f_H)$
            if $f_H$ is redundant, then approximate $f_H$ in $H$
        Update $EA$
}
```

Fig. 5. Probabilistic approximation algorithm

the original circuit. We also compute the initial SMA of each fault and the fault probabilities, which are equivalent in the three circuits. Then, we enter the approximation loop. In each iteration, we select for approximation the fault f_A with the minimum probability. This fault can result either in an under-approximation or in an over-approximation. Depending on the type of fault, we add it to A_F or A_H, respectively. Then we compute the new fault probabilities for all possible faults and update the total probability estimation. Note that if the fault results in an under-approximation, the second case does not apply and only one of the terms in the first case needs to be computed because the other does not vary. Similarly, if the fault results in an under-approximation, the third case does not apply and only the other term in the first case needs to be computed. Finally, the approximation is performed and we update the SMA of all faults in the approximate circuit. Along this process we also eliminate possible redundancies that the approximation may have created elsewhere. The process is repeated until the EP target and the EA target are met.

IV. Evolutionary design of approximate logic circuits

The proposed method is based on CGP, in which a circuit is represented as a fixed-sized cartesian grid of $n_r \times n_c$ nodes interconnected by a feed-forward network (see Figure 6). Node inputs can be connected either to one of n_i primary inputs or to an output of a node in preceding L columns. Each node has a fixed number of inputs n_a (usually $n_a = 2$) and can perform one of the logic functions from a predefined set Γ. Each of n_o primary circuit outputs can be connected either to a primary input or to a node's output. The area and delay of the circuit can be constrained by changing the grid size and the L-back parameter.

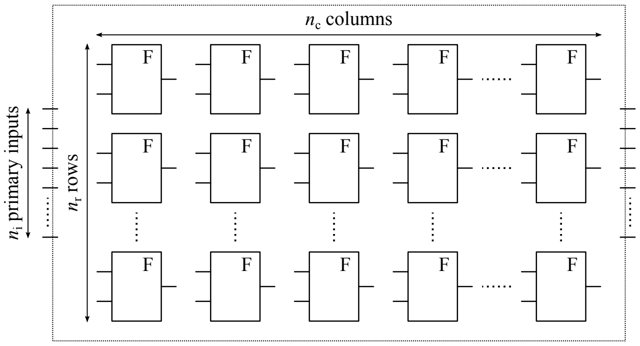

Fig. 6. Cartesian genetic programming schema.

While the search is conducted at the level of genotypes (arrays of integers representing the circuit), the fitness function evaluates phenotypes (circuits established according to the genotypes). The actual encoding is as follows: The primary inputs and the outputs of nodes are labeled $0 \dots n_c \cdot n_r + n_i - 1$ and considered as addresses which connections can be fed to. In the genotype, each two-input node is then encoded using three integers (an address for the first input; an address for the second input; a node function). Finally, for each primary output, the genotype contains one integer specifying the connection address. The genotype size is $(n_a + 1)n_r n_c + n_o$ genes (integers). While the genotype is of fixed length, the size of the phenotype depends on the number of inactive nodes, i.e. nodes whose output is not used by any other node or primary output. Since the inactive nodes have no influence on the phenotype, there are individuals with different genotypes but the same phenotypes.

An example of a CGP individual with its chromosome can be seen in Figure 7. It has three inputs, one output and three active nodes.

CGP uses a simple mutation based $(1 + \lambda)$ evolutionary strategy as a search mechanism. The population size $1 + \lambda$ is mostly very small, typically, $\lambda = 4$. The maximum number of generations created in a single run is N_g. The initial population

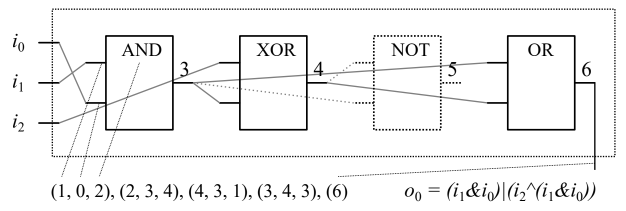

Fig. 7. CGP individual example.

is constructed either randomly (in the case of evolutionary design) or by mapping of a known solution to the CGP chromosome (in the case of evolutionary optimization). In each generation, the best individual is passed to the next generation unmodified along with its λ offspring individuals created by means of a point mutation operator. In case more individuals with the best fitness exist, a randomly selected one is chosen. The mutation rate m is usually set to modify up to 5 % randomly selected genes. The role of mutation is significant in CGP (see detailed analysis in [43], [44]).

Based on our previous experiences, we decided to use a multi-stage single-objective approach with constrained resources to obtain desired approximations. The fitness function f_{under} used to find under-approximations is defined as follows:

$$f_{\mathrm{under}} := \begin{cases} f_{\mathrm{hamm}}^{\mathrm{max}} + (f_{\mathrm{area}}^{\mathrm{max}} - f_{\mathrm{area}}) & \text{if } f_{\mathrm{hamm}} = 0, \\ f_{\mathrm{hamm}}^{\mathrm{max}} - f_{\mathrm{hamm}} & \text{if } f_{\mathrm{off}} = 0, \\ f_{\mathrm{off}}^{\mathrm{max}} - f_{\mathrm{off}} & \text{otherwise,} \end{cases} \tag{1}$$

where f_{hamm} is the total Hamming distance between the outputs generated by the candidate solution and the original circuit for all possible input combinations, $f_{\mathrm{hamm}}^{\mathrm{max}} = n_{\mathrm{o}} 2^{n_{\mathrm{i}}}$, f_{area} is the area of the circuit, $f_{\mathrm{area}}^{\mathrm{max}}$ is the maximum area according to chosen number of rows n_{r} and columns n_{c}, f_{off} is the number of $0 \to 1$ errors for all possible input combinations and $f_{\mathrm{off}}^{\mathrm{max}}$ is the number of zeros in the truth table of the original circuit. All individuals with fitness $f_{\mathrm{under}} \geq f_{\mathrm{off}}^{\mathrm{max}}$ represent a valid under-approximation.

Similarly, the fitness function f_{over} used to find over-approximations is defined as follows:

$$f_{\mathrm{over}} := \begin{cases} f_{\mathrm{hamm}}^{\mathrm{max}} + (f_{\mathrm{area}}^{\mathrm{max}} - f_{\mathrm{area}}) & \text{if } f_{\mathrm{hamm}} = 0, \\ f_{\mathrm{hamm}}^{\mathrm{max}} - f_{\mathrm{hamm}} & \text{if } f_{\mathrm{on}} = 0, \\ f_{\mathrm{on}}^{\mathrm{max}} - f_{\mathrm{on}} & \text{otherwise,} \end{cases} \tag{2}$$

where f_{on} is the number of $1 \to 0$ errors for all possible input combinations and $f_{\mathrm{on}}^{\mathrm{max}}$ is the number of ones in the truth table of the original circuit. All individuals with fitness $f_{\mathrm{over}} \geq f_{\mathrm{on}}^{\mathrm{max}}$ represent a valid over-approximation. One can observe that since all possible input vectors have to be generated, the approach is not scalable. In order to speed up the design, the parallelism at various levels (data, thread, process) can be introduced [27]. In practice, it is applicable for circuits containing less than approximately 20 inputs and 200 gates. More complex circuits can be optimized by introducing formal methods, e.g. SAT solvers or Binary Decision Diagrams (BDD), however, an initial fully working solution is needed in this case [45].

V. Experimental Results

A. Experimental setup

The experiments were conducted with two groups of benchmarks extracted from LGSynth93 set. The first group was intended to compare the results of both probabilistic and evolutionary approaches. Therefore, the size of the circuits in this group was limited to the capabilities of the evolutionary approach. With the second group of benchmarks the goal was to show the efficiency of the probabilistic approach for other circuits and to compare it with other approaches. Benchmarks b12, rd73 and t481 were selected for the first group, and apex3, apex4, m3, misex3, table3 and table5 for the second. The original version of each benchmark was obtained by synthesizing the circuit with Synopsys using the Nangate15nm synthesis library [46]. Table I shows the size of each benchmark as provided by the synthesis tool both in the number of cells and the area, as well as the number of primary inputs (PIs) and outputs (POs).

TABLE I
Synthesis results for selected benchmarks

Benchmark	#PIs	#POs	#cells	area
apex3	54	50	799	192.77
apex4	9	18	1191	279.77
b12	15	9	58	12.93
m3	8	16	205	46.55
misex3	14	14	1285	290.14
rd73	7	3	20	5.90
t481	16	1	32	6.98
table3	14	14	478	116.59
table5	17	15	420	103.61

The generation of approximate logic circuits for the experiments was done in the following way. For the probabilistic approach, a set of arbitrary error targets was set for each circuit, covering a significant range of cases between conventional TMR (full protection) and trivial approximation (no redundant logic). Each error target generated a pair of approximate circuits (over-approximation and under-approximation), which were resynthesized in order to remove logic constants. Each pair of approximate circuits was combined with the original circuit to build an error masking schema.

With respect to the evolutionary approach, a large set of approximate circuits was generated for each benchmark. Table II summarizes the CGP parameters whose values were set up as recommended in the literature [26].

TABLE II
CGP parameters

Parameter	b12	rd73	t481
n_{i}	15	7	16
n_{o}	9	3	1
n_{c}	6	5	5
n_{r}		$1 \dots 10$	
L	6	5	5
Γ		all 2-input gates	
λ		4	
m		5 %	
N_{g}	2000000	2000000	1000000

For each configuration of the CGP grid (as n_{r} varies from 1 to 10), a total of 100 over-approximations and 100 under-approximations were generated. In total, 1000 approximate circuits of each type were generated for each benchmark. According to the fundamentals of the proposed technique, any over-approximation can be combined with any under-approximation to conform a valid error masking scheme. Therefore, there are 10^6 possible solutions to test for each benchmark. Testing the whole set of solutions would take too much time, and therefore it was studied if there was any representative data that allowed to select the best solutions in terms of error masking capabilities. This is discussed in section V-B. Figures 8 and 9 show, in the form of box plots, a statistical analysis of multiple CGP runs for selected circuits evolved as the under-approximations and over-approximations. The box plots give the Hamming distances obtained for increased amount of resources available for the implementation. A clear trade-off between the Hamming distance (quality) and the area can be observed.

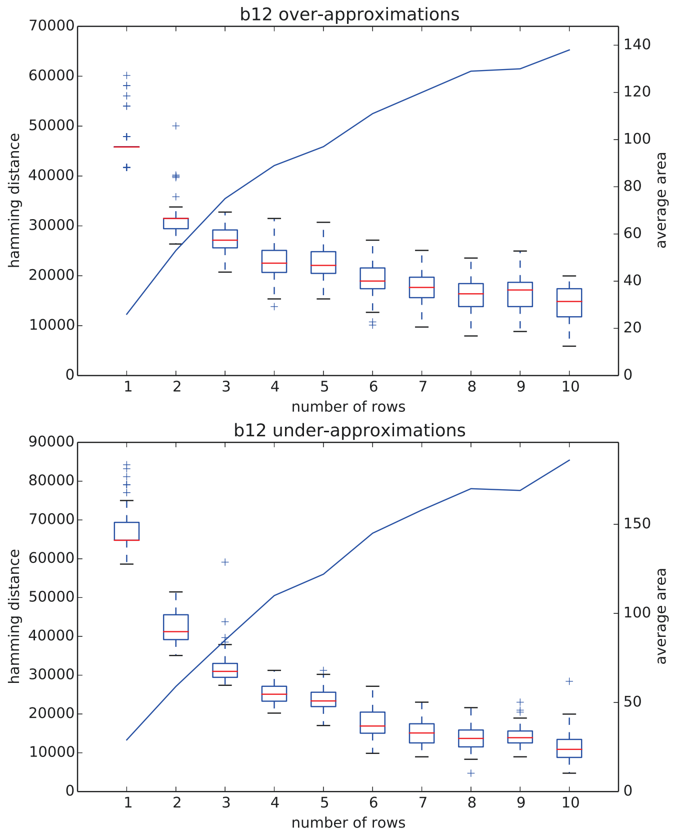

Fig. 8. Statistical results for b12 approximations evolved by CGP.

Once approximate circuits were generated, masking schemas were built for testing. Voters were placed at the output of circuits, and the list of stuck-at faults was generated for each circuit. This list included all faults on every input of each gate in the circuit, plus the faults on the outputs of the circuit before the voter. This allowed to introduce simple voters, as there is full control of fault injection points.

For each error masking schema under test, a fault simulation with random input vectors was performed by means of the parallel simulator HOPE [47]. A total of 50000 randomly generated input vectors were applied for each design under

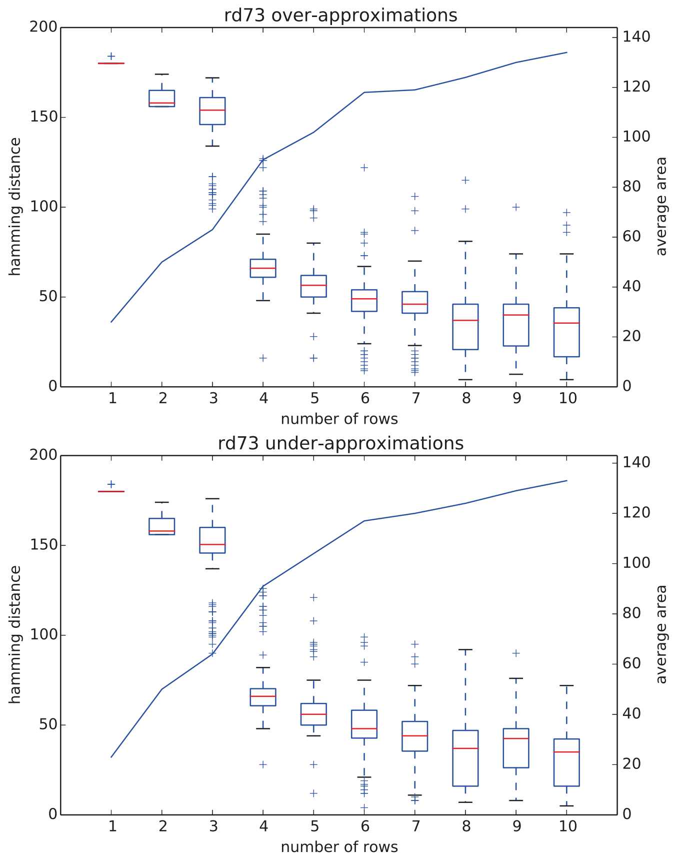

Fig. 9. Statistical results for rd73 approximations evolved by CGP.

test, and all faults in the list previously generated were tested for each input vector. The total error probability was computed as the average number of faults detected per input vector, divided by the size of the fault list. For simplicity, all faults were considered equally likely.

At the time of estimating the probability of circuit failure, it must be taken into account that the number of faults hitting a circuit is correlated with the circuit area. Thus, as the area increases by using larger approximations, the fault probability increases as well. However, all faults in the approximate circuits are either masked or detectable, so that the probability of circuit failure always decreases as the quality of the approximation increases.

B. Selection of candidate Approximate Logic Circuits from evolutionary approach

As stated before, 1000 approximate circuits of each type were generated with the evolutionary approach for the experiments, which made a total of 10^6 combinations to test. For the first benchmark (b12) an exhaustive analysis was performed. The whole process is very time consuming, therefore data collected for b12 were studied in order to properly select the most promising candidates for the rest of the benchmarks. The objective was to find the combination with the highest error masking rate. The more functionally similar are the approximate circuits with respect to the original circuit, the more protection against faults is achieved. Therefore, approximate circuits with low Hamming distance compared with the original circuit are good candidates, in principle. To validate this hypothesis, the correlation between the sum of Hamming distances of both approximate circuits and the experimental error probability was computed. The results are shown in Table III, grouped according to the size of the configuration matrix for both under- and over-approximate circuits. The results show that there is a significant correlation between both metrics. The average correlation index is 0.831. Therefore, for the rest of the benchmarks only the circuits with a Hamming distance below the average of each group were selected for experiments.

C. Comparison between probabilistic and evolutionary approaches

First, the results of experiments aimed to compare both techniques are shown here. Fig. 10 graphically represents the tradeoff between error probability and area overhead for several solutions found by using either the probabilistic or the evolutionary approach. This probability is related to the number of faults in the original circuit in order to take into account the area increase. For the latter technique, only the cases with the best error masking rate for each possible combination in the sizes of under-approximate and over- approximate circuits are represented. The same applies for Fig. 11 and 12 with respect to rd73 and t481 benchmarks, respectively.

Analyzing the results in Fig. 10, it can be seen that the evolutionary approach achieves in general slightly better results than the probabilistic one for b12 benchmark. This is reasonable, because evolutionary approach can explore a much larger range of solutions, although at a much larger computational cost. However, the probabilistic approach can still obtain good solutions, close to the evolutionary approach.

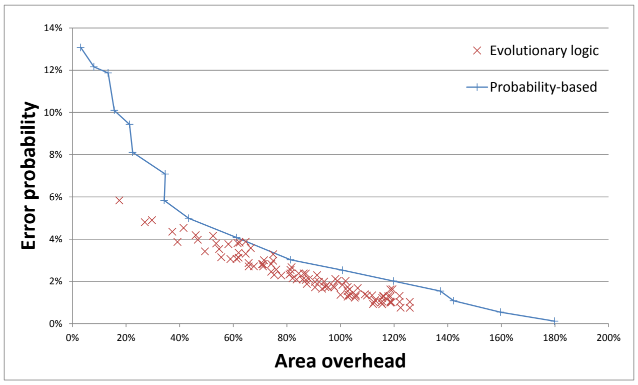

Fig. 10. b12 Simulation Results

Results for rd73 benchmark are shown on Fig. 11. This is an example of a circuit with a high degree of binateness, which means that small expansions on either the on-set or the off-set have a high cost in terms of resources. This leads to sub-optimal solutions with overheads greater than 200 % in both approaches, which are uninteresting. On the other hand, under the 200 % overhead limit the same tendency is observed. The evolutionary approach produces slightly better solutions than the probabilistic approach, but with much more computational effort.

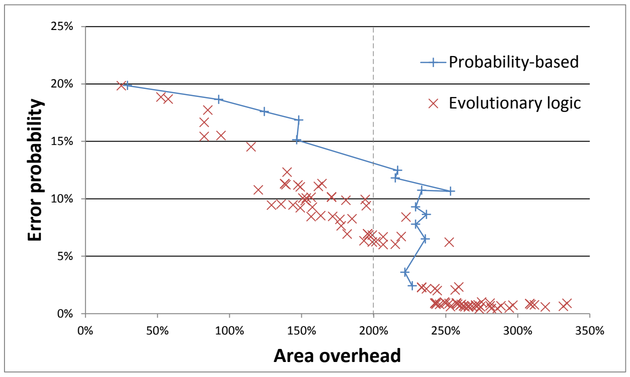

Fig. 11. rd73 Simulation Results

With respect to t481 benchmark (Fig. 12), it can be observed that the probabilistic approach presents more scalability than the evolutionary one. This is due to the fact that t481 is a circuit with just one output with high onset probability. Under such conditions, the evolutionary approach tends to generate onset circuits which behave like logic constants due to the limited size of the configurable logic array, thus limiting both area overhead and error masking rate. On the other hand, the probabilistic approach is based on gradually degrading the logic function of the circuit, which allows to reach more robust solutions in the region close to conventional TMR.

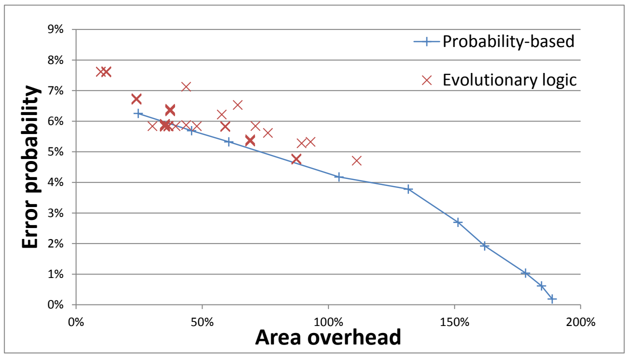

Fig. 12. t481 Simulation Results

D. Comparison with other approaches

A second group of experiments was performed with the aim of showing the applicability and scalability of the probabilistic approach for larger circuits. In addition, the results from this group were compared with the sub-circuit resynthesizing method recently proposed in [11]. The results of the experiments are shown in Fig. 13. The graphics show for each benchmark the tradeoff between the EP improvement, i.e., the improvement in the EP with respect to the original

TABLE III
ERROR MASKING RATE VS. HAMMING DISTANCE CORRELATION INDEXES

Over/Under	1	2	3	4	5	6	7	8	9	10
1	0.700	0.686	0.739	0.754	0.708	0.707	0.674	0.646	0.688	0.667
2	0.792	0.772	0.803	0.843	0.818	0.840	0.838	0.820	0.845	0.839
3	0.717	0.626	0.653	0.732	0.704	0.766	0.770	0.763	0.786	0.797
4	0.802	0.734	0.820	0.875	0.833	0.867	0.890	0.879	0.887	0.875
5	0.806	0.737	0.799	0.847	0.815	0.849	0.867	0.857	0.868	0.861
6	0.836	0.818	0.871	0.903	0.885	0.900	0.916	0.910	0.914	0.913
7	0.823	0.814	0.861	0.897	0.882	0.897	0.913	0.905	0.911	0.912
8	0.811	0.805	0.853	0.893	0.880	0.896	0.912	0.902	0.911	0.914
9	0.795	0.751	0.795	0.854	0.834	0.867	0.887	0.875	0.888	0.893
10	0.815	0.840	0.865	0.901	0.900	0.913	0.929	0.923	0.930	0.938

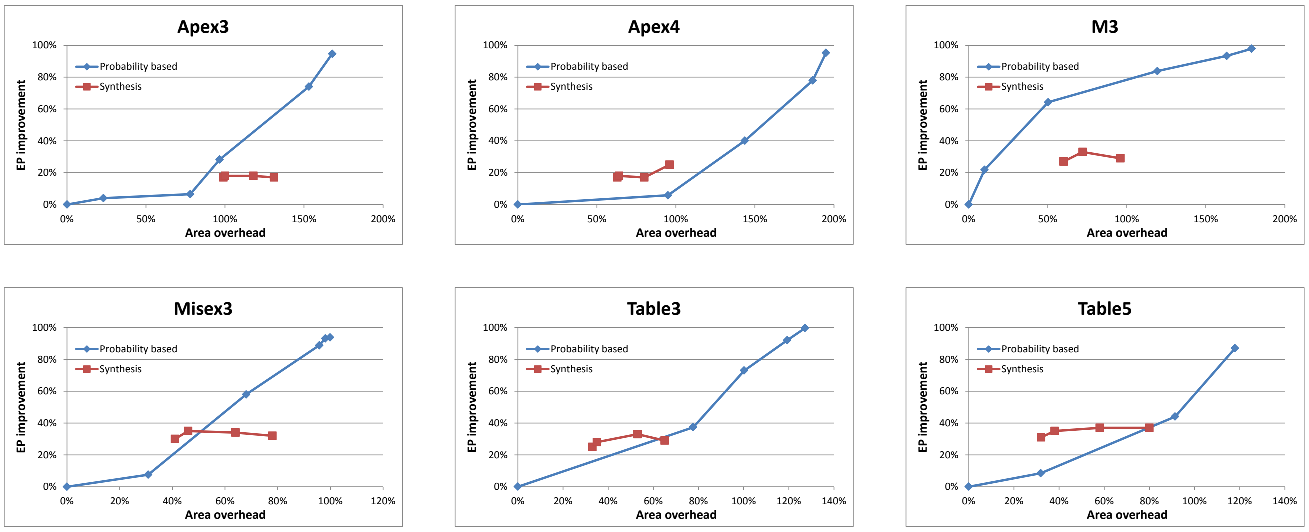

Fig. 13. Simulation results and comparison with synthesis-based approach [11]

unprotected circuit, and the area overhead for different targets. For the probabilistic approach, the additional area due to the approximate circuits and single voters is taken into account. The graphics also include data from [11] for the sake of comparison. It must be noted that these data were obtained with a different synthesis tool and a different technology. Notwithstanding, with the necessary precautions, the results can be used for a general comparison.

The results show that the synthesis approach tends to produce slightly more efficient solutions with respect to the probabilistic approach, although the probabilistic approach can produce better results in some cases. On the other hand, the synthesis approach evinces a very limited scalability, while the probabilistic approach is able to reach any level of error protection, as high as desired. As a matter of fact, the synthesis approach cannot significantly improve the EP by increasing the area overhead.

Finally, it must be noted that a single particle strike can generate a multiple fault in adjacent cells due to charge sharing. This effect is critical in the synthesis approach, as a multiple fault can invalidate the logical masking. However, our approach provides protection against this effect by construction, because the three circuits are built separately. Thus, a multiple fault caused by charge sharing can only affect one of the circuits and the multiple error can be masked at the voter.

VI. CONCLUSIONS

In this work we proposed and compared two different approaches to generate approximate logic circuits for error mitigation using a TMR schema. The probablilistic approach uses a greedy algorithm based on line approximations and dynamic error probability estimations. The evolutionary approach is based on CGP and can generate radically different solutions. The experimental results show that the evolutionary approach is generally able to find slightly better results, but at the expense of a higher computational effort. On the other hand, the probabilistic approach can handle large circuits in an efficient manner. Notwithstanding, the current progress in evolutionary computing techniques suggests that they will be able to process larger circuits in the near future [45].

The two proposed methods are widely scalable and can provide solutions for any required trade-off between reliability and area overhead. This is a major advantage to cover a variety of application scenarios and technologies. In comparison, recently proposed synthesis-based methods can occasionally produce slightly better results but they cannot explore the design space in depth.

References

[1] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305–316, 2005.

[2] G. Neuberger, G. Wirth, and S. O. service), *Protecting Chips Against Hold Time Violations Due to Variability*. Dordrecht :: Springer Netherlands :, 2014. [Online]. Available: http://dx.doi.org/10.1007/978-94-007-2427-3

[3] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro, IEEE*, vol. 25, no. 6, pp. 10–16, 2005.

[4] V. Ferlet-Cavrois, L. W. Massengill, and P. Gouker, "Single event transients in digital cmosâĂŤa review," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 3, pp. 1767–1790, 2013.

[5] M. M. Shulaker, G. Hills, N. Patil, H. Wei, H.-Y. Chen, H.-S. P. Wong, and S. Mitra, "Carbon nanotube computer," *Nature*, vol. 501, no. 7468, pp. 526–530, 2013.

[6] I. Polian and J. P. Hayes, "Selective hardening: Toward cost-effective error tolerance," *IEEE Design & Test of Computers*, no. 3, pp. 54–63, 2010.

[7] B. W. Johnson, *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc., 1988.

[8] K. Mohanram, N. Touba *et al.*, "Partial error masking to reduce soft error failure rate in logic circuits," in *Defect and Fault Tolerance in VLSI Systems, 2003. Proceedings. 18th IEEE International Symposium on*. IEEE, 2003, pp. 433–440.

[9] A. H. El-Maleh and F. C. Oughali, "A generalized modular redundancy scheme for enhancing fault tolerance of combinational circuits," *Microelectronics Reliability*, vol. 54, no. 1, pp. 316–326, 2014.

[10] K. Nepal, N. Alves, J. Dworak, and R. I. Bahar, "Using implications for online error detection," in *Test Conference, 2008. ITC 2008. IEEE International*. IEEE, 2008, pp. 1–10.

[11] A. El-Maleh and K. Daud, "Simulation-based method for synthesizing soft error tolerant combinational circuits," *Reliability, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2015.

[12] S. Krishnaswamy, S. M. Plaza, I. L. Markov, and J. P. Hayes, "Enhancing design robustness with reliability-aware resynthesis and logic simulation," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*. IEEE, 2007, pp. 149–154.

[13] S. Almukhaizim and Y. Makris, "Soft error mitigation through selective addition of functionally redundant wires," *Reliability, IEEE Transactions on*, vol. 57, no. 1, pp. 23–31, 2008.

[14] B. D. Sierawski, B. L. Bhuva, and L. W. Massengill, "Reducing soft error rate in logic circuits through approximate logic functions," *Nuclear Science, IEEE Transactions on*, vol. 53, no. 6, pp. 3417–3421, 2006.

[15] M. Choudhury and K. Mohanram, "Low cost concurrent error masking using approximate logic circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 32, no. 8, pp. 1163–1176, Aug 2013.

[16] M. R. Choudhury and K. Mohanram, "Approximate logic circuits for low overhead, non-intrusive concurrent error detection," in *Design, Automation and Test in Europe, 2008. DATE'08*. IEEE, 2008, pp. 903–908.

[17] I. Gomes, M. Martins, F. Lima Kastensmidt, A. Reis, R. Ribas, and S. Novales, "Methodology for achieving best trade-off of area and fault masking coverage in atmr," in *Test Workshop - LATW, 2014 15th Latin American*, March 2014, pp. 1–6.

[18] H. Xie, L. Chen, R. Liu, A. Evans, D. Alexandrescu, S.-J. Wen, and R. Wong, "New approaches for synthesis of redundant combinatorial logic for selective fault tolerance," in *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International*, July 2014, pp. 62–68.

[19] A. Sanchez-Clemente, L. Entrena, M. Garcia-Valderas, and C. Lopez-Ongil, "Logic masking for set mitigation using approximate logic circuits," in *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, June 2012, pp. 176–181.

[20] A. Sanchez-Clemente, L. Entrena, and M. Garcia-Valderas, "Error masking with approximate logic circuits using dynamic probability estimations," in *On-Line Testing Symposium (IOLTS), 2014 IEEE 20th International*, July 2014, pp. 134–139.

[21] R. Drechsler, *Evolutionary Algorithms for VLSI CAD*. Boston: Kluwer Academic Publishers, 1998.

[22] E. Larsson, *Introduction to Advanced System-on-Chip Test Design and Optimization*. Springer, 2005.

[23] M. A. Trefzer and A. M. Tyrrell, *Evolvable Hardware: From Practice to Application*. Springer, 2015.

[24] M. Garvie and A. Thompson, "Scrubbing away transients and jiggling around the permanent: long survival of fpga systems through evolutionary self-repair," in *Proc of the 10th IEEE International On-Line Testing Symposium IOLTS 2004*, 2004, pp. 155–160.

[25] J. Miller and P. Thomson, "Cartesian genetic programming," in *Genetic Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2000, vol. 1802, pp. 121–132. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-46239-2_9

[26] J. F. Miller, Ed., *Cartesian Genetic Programming*, ser. Natural Computing Series. Springer Verlag, 2011. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=7299

[27] R. Hrbacek and L. Sekanina, "Towards highly optimized cartesian genetic programming: From sequential via simd and thread to massive parallel implementation," in *GECCO '14 Proceedings of the 2014 conference on Genetic and evolutionary computation*. Association for Computing Machinery, 2014, pp. 1015–1022.

[28] Z. Vasicek and L. Sekanina, "Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware," *Genetic Programming and Evolvable Machines*, vol. 12, no. 3, pp. 305–327, 2011.

[29] ——, "Evolutionary approach to approximate digital circuits design," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, 2015. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=10406

[30] R. Hrbacek, "Parallel multi-objective evolutionary design of approximate circuits," in *GECCO '15 Proceedings of the 2014 conference on Genetic and evolutionary computation*. Association for Computing Machinery, 2015, pp. 687–694.

[31] M. Iyer, M. Abramovici *et al.*, "Fire: a fault-independent combinational redundancy identification algorithm," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 4, no. 2, pp. 295–301, 1996.

[32] L. Entrena, K.-T. Cheng *et al.*, "Combinational and sequential logic optimization by redundancy addition and removal," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 14, no. 7, pp. 909–916, 1995.

[33] S.-C. Chang, M. Marek-Sadowska, and K.-T. Cheng, "Perturb and simplify: multilevel boolean network optimizer," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, no. 12, pp. 1494–1504, 1996.

[34] H. Kim and J. Hayes, "Realization-independent atpg for designs with unimplemented blocks," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 290–306, Feb 2001.

[35] F. Brglez, "On testability of combinational networks," in *IEEE International Symposium on Circuits and Systems*, 1984.

[36] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT-probabilistic estimation of digital circuit testability," in *Proceeding of International Symposium on Fault-Tolerant Computing*, Jun. 1985, pp. 220–225.

[37] S. Chakravarty and I. Hunt, H.B., "On computing signal probability and detection probability of stuck-at faults," *Computers, IEEE Transactions on*, vol. 39, no. 11, pp. 1369–1377, Nov 1990.

[38] S. Jain and V. Agrawal, "Statistical fault analysis," *Design Test of Computers, IEEE*, vol. 2, no. 1, pp. 38–44, Feb 1985.

[39] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *Computers, IEEE Transactions on*, vol. 100, no. 6, pp. 668–670, 1975.

[40] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random pattern testability," *Computers, IEEE Transactions on*, vol. 100, no. 1, pp. 79–90, 1984.

[41] S.-C. Chang, W.-B. Jone, and S.-S. Chang, "Tair: Testability analysis by implication reasoning," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 1, pp. 152–160, 2000.

[42] K. Parker and E. McCluskey, "Analysis of logic circuits with faults using input signal probabilities," *Computers, IEEE Transactions on*, vol. C-24, no. 5, pp. 573–578, May 1975.

[43] J. F. Miller and S. L. Smith, "Redundancy and computational efficiency in cartesian genetic programming," *IEEE Trans. Evolutionary Computation*, vol. 10, no. 2, pp. 167–174, 2006.

[44] B. W. Goldman and W. F. Punch, "Analysis of cartesian genetic programming's evolutionary mechanisms," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 359–373, 2015.

[45] Z. Vasicek and L. Sekanina, "How to evolve complex combinational circuits from scratch?" in *2014 IEEE International Conference on Evolvable Systems Proceedings*. Institute of Electrical and Electronics Engineers, 2014, pp. 133–140. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=10673

[46] "Nangate freepdk15 open cell library," 2014. [Online]. Available: http://www.nangate.com/?page_id=2328

[47] H. K. Lee and D. S. Ha, "Hope: an efficient parallel fault simulator for synchronous sequential circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, no. 9, pp. 1048–1058, Sep 1996.