2018 30th International Conference on Microelectronics (ICM)

Approximation-Conscious IC Testing

Mahmoud Masadeh, Osman Hasan, and Sofiène Tahar
Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada
Email:{m_masa,o_hasan,tahar}@ece.concordia.ca

***Abstract*—Approximate computing is a nascent energy-efficient computing paradigm for error-tolerant applications. However, the approximate nature of these circuits makes their testing phase quite challenging. Similarly, partial testing of ICs based on a reduced fault list is adapted to exclude some test patterns for manufacturing defects tolerated by design approximation. To streamline these processes and thus reduce yield loss and test cost, and based on distinct subsets of test patterns and fault coverage, we propose an approximation-conscious multi-level IC test flow, which classifies the output of the test process to be either: (1) a "good" defect-free IC, (2) 7 different levels of "good-enough" partially-passed approximate ICs, or (3) a "bad" rejected IC.**

I. Introduction

The fact that the transistor size is reaching the physical limits, i.e., 10nm by the year 2021 [1], has led to the emergence of a few challenges related to reliability, complex manufacturing process, and high testing cost. To ensure high reliability of Integrated Circuits (ICs) during the operational life time, fault tolerant designs are used, i.e., BIST (built-in self-test) and BISR (built-in self-repair), which introduce extra area, timing delays and power consumption. Moreover, Process-Voltage-Temperature (PVT) variations require adding extra guard bands, i.e., low clock frequency and high supply voltage, to ensure that the manufactured ICs operate properly [2]. Recently, it has been proposed to have ICs that do not adapt fault tolerance designs [3]. Such hardware usually produces erroneous outputs which can be tolerated in error-resilient applications. This designing paradigm is called *computing on unreliable hardware*, which is adapted by approximate computing (AC) [4], where the AC aims to reduce circuit complexity to minimize area, delay and power consumption.

Functional equivalence between specifications and circuit implementation is relaxed in approximate circuits, to improve efficiency by violating accuracy, in error resilient applications [5]. This nascent research direction is influenced by the development in two domains, 1) *Low power circuit design*: AC utilizes error resiliency in some application. Therefore, computation accuracy is introduced as a new design metric, to trade performance vs power consumption, 2) *Specific properties of ICs in the nano-scale era:* recent fabrication technology shows reliability and uncertainty constraints, therefore AC is a solution for energy efficient systems on unreliable platforms.

Traditional design and verification techniques are not directly applicable to approximate computing [5]. Moreover, the AC design paradigm requires integration in the *IC design flow*, i.e., synthesis, verification and simulation. Therefore, several methods have been proposed to automate the whole process of designing complex approximate circuits, such as SALSA [6] and ABACUS [7].

A defected IC will have an unintended difference between the implemented hardware and the intended design [8], emerged from the manufacturing process that was not originally defined in the design circuit, e.g., open and bridge defects. The probability of defects in ICs increases with decreased feature size [9]. *Failures* are the physical manifestation of the defect, and *fault* is the mathematical model that describes the behavior of this failure such as Stuck-At-Zero (SA0), and Stuck-At-One (SA1) fault. Fault abstraction reduces the complexity since as many defects have the same fault behavior. *Fault Model* is a collection of faults with similar properties; such as Stuck-At-Fault (SAF) model that includes SA0 and SA1 faults. Other fault models include Bridging fault model, Stuck Open Fault model, Transistor Stuck-Open Fault model, and Transistor Stuck-Short Fault model. *Fault models should accurately reflect the behavior of defects; as they are used for generating and evaluating test patterns* [10]. In this work, without loss of generality, we target SAF model.

Just like synthesis, verification and simulation, it is required to integrate approximate computing into the *IC test flow*, i.e., test generation, application, and evaluation. However, the research in this topic still scarce. The work in [11] aimed to reduce test cost and time of exact circuits, through generating test patterns for the most vulnerable circuit elements, which is called *approximate testing*. However, the work [11] did not target approximate circuits. In [12], the authors identified all faults, which do not violate the *worst-case error* (maximum error distance) metric for a manufactured approximate circuit. These identified faults are removed from the fault list, to increase the yield. Identifying faults that are violating the *worst-case error* is a straightforward task compared to the identification of the faults that violate *mean error metrics* (e.g., mean error distance and mean relative error distance). There are several application dependent *error metrics* used in approximate computing to quantify approximation errors and evaluate design accuracy [13], such as:

- Error Rate (ER): The percentage of erroneous outputs among all outputs.
- Error Distance (ED): The arithmetic difference between the exact result and approximate result.
- Maximum Error Distance (*worst-case error*): The maximum error distance among all approximate outputs.
- Mean Error Distance (MED): The average of all EDs for a set of outputs obtained by applying a set of inputs.

978-1-5386-8167-1/18/$31.00 ©2018 IEEE

- Relative Error Distance (RED): The ratio of ED to the exact output.

In this work, we propose an algorithm to identify approximation-redundant faults, which do not violate the *mean error distance (MED)*, so they can be dropped from the fault list. For non-approximation faults, we identify the test patterns that may have a significant impact on the error metric. Moreover, we propose an approximation-conscious multi-level IC test flow, which classify the output of the test process to be either: (1) a "good" defect-free IC, (2) 7 different levels of "good-enough" partially-passed approximate ICs, or (3) a "bad" rejected IC.

The rest of this paper is organized as follows. Section II explains our proposed multi-level IC test flow. The proposed fault classification algorithm and a 2-bit binary adder application are explained in Sections III and IV, respectively. Finally, some conclusions are drawn in Section V.

II. Proposed Multi-Level IC Test Flow

The *post-manufacturing* testing process is a main step in the VLSI design cycle. While the manufacturing cost of transistors is decreasing, testing cost is fixed and is becoming dominating in the low technology nodes. This necessitate reducing IC test length and time. *Functional testing* of a circuit with P inputs, requires 2^P test patterns, e.g., 2^{64} test patterns for a 32-bits binary adder, which is quite impractical for real circuits. Therefore, *structural testing* [8] based on fault models, has been proposed to reduce test complexity, based on developing technology independent fault models and test algorithms.

To the best of our knowledge, there is no previous work in IC test flow that integrates both *circuit approximation* and *approximate testing* while considering various error metrics, such as *error rate* (ER) and *mean error distance* (MED). The main objective of *approximate testing* is to eliminate test patterns from the fault list with the motivation of test cost reduction. Our proposed approximation-conscious IC test flow, shown in Figure 1, allows us to integrate approximate circuits in the IC test flow, i.e., test generation, application, and evaluation. The proposed approach encompasses both exact and approximate designs. We mainly propose a classification of the tested circuits into seven different levels based on the applied test patterns and the required fault coverage:

(***L1***) When an exact circuit is tested with a full list of test patterns (TP1) for SAF model, and 100% fault coverage (FC) is required and met for very critical application, then a *Good-IC* is obtained, and the IC is rejected for FC<100%. However, in approximate computing we have the flexibility to make use of these defective components with high FC to enhance the final yield. Thus, based on the level of acceptable quality (FC), we intentionally consider the ICs with a FC<100% as *an approximate ICs manufactured without planning* for non-critical applications, which have not been optimized for power reduction. We call such ICs *L1-ICs*. A similar classification, i.e., *threshold testing* was also proposed in [14] where the idea is to accept defected ICs as second class ICs. However, they did not classify them as approximate ICs as is the case with our proposed approach.

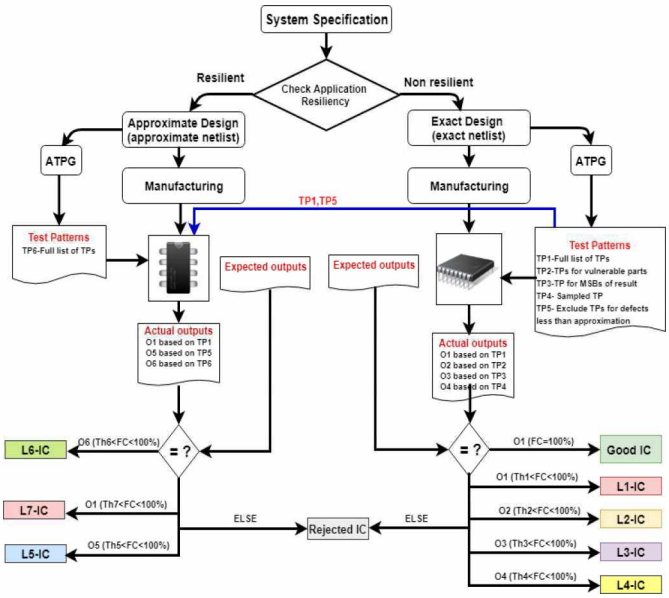

Figure 1: Proposed Approximation-Conscious IC Test Flow

(***L2***) An approximate *L2-IC* is obtained when we check an exact circuit by generating test patterns for the most vulnerable circuit components (TP2), which provide a trade-off between quality and test complexity [11] and the actual output of testing (O2) matches the expected fault coverage threshold (Th2).

(***L3***) For exact circuit with known design structure, i.e., arithmetic circuits, we only generate test patterns (TP3) for testing the most significant bits (MSBs) to reduce test length. If the actual output (O3) matches the expected fault coverage threshold (Th3), then the IC is accepted as an approximate circuit called *L3-IC*.

(***L4***) Based on a fault sampling mechanism, we randomly pick a subset of faults from the set of all faults (sampled-faults $\ll$ all-faults) and generate test patterns for them (TP4) only. If the actual output of testing (O4) matches the expected fault coverage threshold (Th4), then the IC is accepted as an approximate circuit, called *L4-IC*.

(***L5***) We test the approximate circuit, considering the exact design as our reference, for selected faults that violate a specific error metric constraint, using TP5 which is a subset of TP1. There is no need to test the given circuit for manufacturing defects that do not violate approximation error. Thus, both yield loss and test cost would reduce. If a specific fault coverage is met, then the design is accepted as an approximate circuit, called *L5-IC*. The authors of [12] describe a similar technique that considers the *worst-case error* as an error metric. In Section III, we propose an algorithm to identify faults that violate *error rate* and *mean error distance*.

(***L6***) We also test the approximate circuit, based on a full list of test patterns (TP6), generated based on the approximate design. This test is used to detect manufacturing defects only, where approximation errors are not considered. If the actual output (O6) matches the expected fault coverage then the

design is accepted as an approximate circuit, i.e., *L6-IC*, without any manufacturing defects.

(*L7*) Finally, we test the approximate circuit, while considering the exact design as the reference, based on the full list of test patterns (TP1). If the fault coverage is within a specified threshold, then the design is accepted as an approximate circuit, which is called *L7-IC*.

In the proposed IC test flow, we consider the SAF model only and use the Fault Coverage metric for evaluation. However, the proposed multi-level test flow is applicable for other fault models and different evaluation metrics for various ICs application domains.

III. Fault Classification Algorithm

The authors of [12] described a technique to eliminate faults that do not violate the *worst-case error* metric from the fault list. Alternatively, we propose an algorithm to remove approximation-redundant faults, which do not violate the *mean error distance* (MED) from the fault list. Our proposed algorithm is applicable to other error metrics, such as ER, NMED, RED and MRED.

The inputs to fault classification algorithm (Algorithm 1) include a fault list (Faults), exact circuit design (C), a list of faulty circuits (C_f), and error metric (MED). For each fault f in the fault list, MED is evaluated for that faulty circuit (Line 3) by applying all input combinations to the circuit. Then, approximation-redundant faults, which do not violate the MED are removed from the fault list (Line 5). For the non-approximation faults, input patters that cause the error metric to be violated are found (Lines 13-21). Searching for the input patterns can be sequential (from 1 to 2^P). However, a preprocessing step (Line 7) can quickly sort the input patterns based on their maximum error distance, such that the probability of getting a satisfying solution will be higher in a more efficient manner. The algorithm returns back a reduced list of test patterns for the faults, which violate the MED accuracy metric.

Algorithm 1 Fault Classification based on MED

```
1:  procedure FAULT CLASSIFICATION(Faults, C ,C_f, MED)
2:      for each f ∈ Faults do
3:          MED_f ← EvaluateMED(C_f)        ▷ C_f is the faulty circuit with fault f
4:          if MED_f ≤ MED then
5:              Faults_new ← Faults − f      ▷ fault f is approximation-redundant
6:          else                             ▷ fault f violates MED metric
7:              PreProcessing()              ▷ Quickly Sort Input patterns
8:              Patterns = Find_Input_Patterns(f, Circuit,P)
9:          end if
10:     end for
11:     return Patterns
12: end procedure
13: function FIND_INPUT_PATTERNS(f, Circuit,P)
14:     for each i ∈ 2^P do
15:         MED_fi ← EvaluateMED(C_fi)       ▷ MED_fi is MED for faulty circuit C_f excluding input pattern i
16:         if MED_fi < MED then
17:             TestPatterns ← TestPatterns + i   ▷ Input i violates the error metric
18:         end if
19:     end for
20:     return TestPatterns
21: end function
```

IV. Application

Functional approximation is used mainly in arithmetic circuits, like binary adders and multipliers. In this section, an *N-bit ripple carry adder (RCA)* is introduced to show the benefits of *structural testing* compared to *functional testing*.

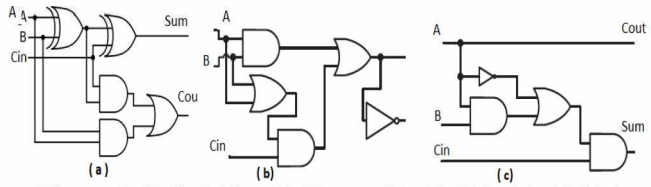

Figure 2: Full Adder (a) Exact, (b) AMA2, (c) AMA4

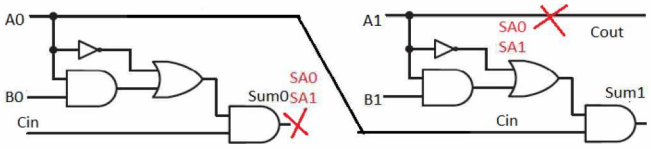

Figure 3: 2-bits AMA4-based Binary Adder with SAF

A 1-bit exact FA, an approximate mirror adder 2 (AMA2), and an approximate mirror adder 4 (AMA4) are shown in Figures 2(a), 2(b) and 2(c), respectively [15]. All cells have the same I/O interface with a different internal structure. The approximated cells have fewer gates, smaller size, shorter critical path and less power consumption. These benefits are acquired by compromising the accuracy. Moreover, fewer gates and internal branches lead to less fault locations. Thus, testing an approximate IC requires a reduced number of test patterns with a reduced test cost and time. Thus, testing of approximate circuits is expected to be faster and cheaper than testing the exact circuits. The number of *fault sites* for FA cells in Figures 2(a), 2(b), and 2(c) is 16, 14 and 10, respectively. The number of faults sites for the exact, AMA2 and AMA4 FAs based N-bit ripple carry adder (RCA) is $15N + 1$, $13N + 1$, and $9N + 1$, respectively. The number of test patterns is *double* the number of fault sites, i.e., fault sites may have SA0 or SA1 fault. On the other hand, the number of test patterns for exhaustive simulation is 2^{2N+1} for N-bit RCA. Since this is a huge number, we use the test patterns based on SAF model.

We use a 2-bit RCA based on AMA4, as depicted in Figure 3, to show the effectiveness of our proposed algorithm. The fault list of the circuit includes 38 faults. For illustration purposes, we use Stuck-At-One (*SA1*) and Stuck-At-Zero (*SA0*) faults, at both *Cout* and *Sum0* outputs. We classify faults to be either approximation-redundant or non-approximation, while considering only the *mean error distance* metric, which have a strong correlation with the *error rate* metric. Approximation-

Table I: Truth Table for SAF at Cout and Sum0 for 2-bit Approximate Adder, and Various Error Metrics

Exact		Approximate		SA0@Cout		SA1@Cout		SA0@Sum0		SA1@Sum0	
Inputs	Output	Output	ED	Result	ED	Result	ED	Result	ED	Result	ED
00000	000	000	0	000	0	100	4	000	0	001	1
00001	001	001	0	001	0	101	4	000	1	001	0
00010	001	000	1	000	1	100	3	000	1	001	0
00011	010	001	1	001	1	101	3	000	2	001	1
00100	001	010	1	010	1	110	5	010	1	011	2
00101	010	010	0	010	0	110	4	010	0	011	1
00110	010	010	0	010	0	110	4	010	0	011	1
00111	011	011	0	011	0	111	4	010	1	011	0
01000	010	000	2	000	2	100	2	000	2	001	1
01001	011	001	2	001	2	101	2	000	3	001	2
01010	011	000	3	000	3	100	1	000	3	001	2
01011	100	001	3	001	3	101	1	000	4	001	3
01100	011	010	1	010	1	110	3	010	1	011	0
01101	100	010	2	010	2	110	2	010	2	011	1
01110	100	010	2	010	2	110	2	010	2	011	1
01111	101	011	2	011	2	111	2	010	3	011	2
10000	010	100	2	000	2	100	2	100	2	101	3
10001	011	101	2	001	2	101	2	100	1	101	2
10010	011	100	1	000	3	100	1	100	1	101	2
10011	100	101	1	001	3	101	1	100	0	101	1
10100	011	100	1	000	3	100	1	100	1	101	2
10101	100	100	0	000	4	100	0	100	0	101	1
10110	100	100	0	000	4	100	0	100	0	101	1
10111	101	101	0	001	4	101	0	100	1	101	0
11000	100	100	0	000	4	100	0	100	0	101	1
11001	101	101	0	001	4	101	0	100	0	101	0
11010	101	100	1	000	5	100	1	100	1	101	0
11011	110	101	1	001	5	101	1	100	2	101	1
11100	101	110	1	010	3	110	1	110	1	111	2
11101	110	110	0	010	4	110	0	110	0	111	1
11110	110	110	0	010	4	110	0	110	0	111	1
11111	111	111	0	011	4	111	0	110	1	111	0
Error Rate			0.59		0.84		0.75		0.72		0.75
Mean Error Distance			0.94		2.44		1.75		1.19		1.13
Worst Case Error			3.00		5.00		5.00		4.00		3.00

redundant faults are dropped-out from the fault list. Non-approximation faults have to be considered while generating the list of test patterns.

Table I shows the truth table for 2-bit RCA. First two columns are the exact inputs ($CinA1A0B1B0$) and outputs ($CoutSum1Sum0$). Column 3 shows approximate output without any SAF, and Column 4 shows the associated error distance (ED), where the maximum obtained value of ED is 3. The remaining columns show different faults at different outputs. Faults with error distance that violate the *worst-case error*, greater than 3, are highlighted (yellow cells) in Table I. Fault SA1@Sum0 does not violate the *worst-case error*. All faults in Table I violate the *mean error distance*. The set of test patterns generated based on Algorithm 1 to detect each fault are shown with the underlined text in Table I. Since there is a correlation between error rate and MED *metrics*, we notice a correlation relationship between *test patterns* to detect faults which violate MED and *test patterns* to detect faults which violate error rate.

V. Conclusion

The emergence of approximate computing has led to the emerging of interesting energy-efficient designs with low delay, and high performance. However, this paradigm requires to be integrated into the IC *design flow* as well as the *test flow*. The integration of approximate computing into test flow has the potential to reduce test cost and yield loss. With this motivation, we proposed an approximation-conscious IC test flow with 7 levels of approximation. The main idea is about forming distinct subsets of test patterns, which ensure high fault coverage with reduced test time. Based on the proposed methodology, the tested ICs can be classified as *Good-IC* with 100% accuracy, an *approximated-IC* based on 7 different levels of approximation, or as a *rejected-IC* with an unacceptable fault coverage. Eliminating the test patterns for manufacturing defects tolerated by approximation is an essential step of this process. For that, we propose an algorithm to classify faults under the *mean error distance* to either approximation redundant and non-approximation faults. Approximation redundant faults can be removed while generating test patterns. Non-approximation faults are used to generate reduced test patterns, which are non-deterministic because there is more than one acceptable answer (list of test patterns) to this problem. The proposed algorithm is validated with a case-study of a 2-bit binary adder using AMA4 FA cells. For future work, we plan to target other fault models other than the SAF model, more evaluation metrics, and various accuracy metrics.

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2017.

[2] A.Bosio, A.Virazel, P.Girard, and M.Barbareschi, "Approximate computing: Design and test for integrated circuits," in *Latin American Test Symposium*, 2017, pp. 1–1.

[3] M. Carbin and S. Misailovic, "Verifying quantitative reliability for programs that execute on unreliable hardware," in *Object-oriented Programming, Systems, Languages, and Applications*, 2013, pp. 33–52.

[4] L. Anghel, M. Benabdenbi, A. Bosio, and E. I. Vatajelu, "Test and reliability in approximate computing," in *International Mixed Signals Testing Workshop*, 2017, pp. 1–6.

[5] L.Sekanina, "Introduction to approximate computing: Embedded tutorial," in *International Symposium on Design and Diagnostics of Electronic Circuits Systems*, 2016, pp. 1–6.

[6] S. Venkataramani, A. Sabne, V. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: Systematic logic synthesis of approximate circuits," in *Design Automation Conference*, 2012, pp. 796–801.

[7] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "ABACUS: A technique for automated behavioral synthesis of approximate computing circuits," in *Design, Automation Test in Europe*, 2014, pp. 1–6.

[8] M. Taouil, M. Masadeh, S. Hamdioui, and E. J. Marinissen, "Post-bond interconnect test and diagnosis for 3-d memory stacked on logic," *IEEE Transactions on Computer-Aided Design*, vol. 34, no. 11, pp. 1860–1872, 2015.

[9] A. J. van de Goor, S. Hamdioui, and Z. Al-Ars, "Tests for address decoder delay faults in rams due to inter-gate opens," in *European Test Symposium*, 2004, pp. 146–151.

[10] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Elsevier, 2006.

[11] I.Wali, M.Traiola, A.Virazel, P.Girard, M.Barbareschi, and A.Bosio, "Towards approximation during test of integrated circuits," in *International Symposium on Design and Diagnostics of Electronic Circuits Systems*, 2017, pp. 28–33.

[12] A. Chandrasekharan, S. Eggersglüß, D. Große, and R. Drechsler, "Approximation-aware testing for approximate circuits," in *Asia and South Pacific Design Automation Conference*, 2018, pp. 239 – 244.

[13] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *Design, Automation Test in Europe*, 2016, pp. 660–665.

[14] Z. Jiang and S. K. Gupta, "An ATPG for threshold testing: obtaining acceptable yield in future processes," in *International Test Conference*, 2002, pp. 824–833.

[15] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design*, vol. 32, no. 1, pp. 124–137, 2013.