Approximation-aware Testing for Approximate Circuits

Arun Chandrasekharan, Stephan Eggersglüß, Daniel Große, Rolf Drechsler

Abstract—A wide range of applications significantly benefit from the Approximate Computing (AC) paradigm in terms of speed or power reduction. AC achieves this by tolerating errors in the design. These errors are introduced into the design either manually by the designer or by approximate synthesis approaches. From here, the standard design flow is taken. Hence, the manufactured AC chip is eventually tested for production errors using well established fault models. To be precise, if the test for a test pattern fails, the AC chip is sorted out. However, from a general perspective this procedure results in throwing away chips which are perfectly fine taking into account that the considered fault (i.e. physical defect that leads to the error) can still be tolerated because of approximation. This can lead to a significant amount of yield loss.

In this paper, we present an approximation-aware test methodology which can be easily integrated into the regular test flow. It is based on a pre-process to identify approximation-redundant faults. By this, we remove all potential faults that no longer need to be tested because they can be tolerated under the given error metric. Our experimental results and case studies on a wide variety of benchmark circuits show a significant potential for yield improvement.

I. INTRODUCTION

The complexity of chips is steadily increasing while, at the same time, the feature sizes are shrinking. Both facts pose major challenges to the design of todays chips. In addition, better energy efficiency and performance become a major concern. A promising solution is the emerging Approximate Computing (AC) design paradigm. The key idea of AC is to trade off correct computation against energy or performance. The good news is that there are many crucial resource-hungry applications (e.g. audio/video processing, learning, big-data analysis) which can tolerate some deviation of the exact result [1]. From the hardware design side, various handcrafted approximate designs have been proposed, ranging from building blocks such as adders, multipliers, etc. to complex configurable CPU architectures [2]. In the context of design automation for AC, also solutions for synthesis [3], verification [4], simulation [5] etc., have been proposed. In this paper, we focus on the last design step of the chip design, i.e. the post production test. In general, the task of manufacturing test is to detect whether a physical defect is present in the chip or not. If yes, the chip will not be shipped to the customer. However, given an approximate circuit and a physical defect, the crucial question is, whether the chip still can be shipped since the defect can be tolerated under approximation. If we can provide a positive answer to this question, this leads to a significant potential for yield improvement.

In this paper, we present an approximation-aware test methodology. To the best of our knowledge this is the first approach considering the impact of design level approximations in post production test. Our methodology does not radically change the test flow, rather it is a pre-process to classical Automatic Test Pattern Generation (ATPG). The key idea is to classify each fault (the logical manifestation of a defect) in approximation-redundant or non-approximation. For this task, we essentially compare the non-approximated (golden) design against the approximated design with an injected fault under the considered error metric constraint. Using formal methods (SAT and variants) as well as structural techniques allow us to classify the fault. For a wide range of benchmarks, we demonstrate the advantage of our approach. We show that, depending on the concrete approximation and error metric (which is driven by the application), a relative reduction of up to 80% in fault count can be achieved.

In summary, this paper makes the following contributions:

- Identification of approximation-redundant faults
- Mapping into a formal fault classification problem
- Pre-process to classical ATPG
- Significant yield improvement potential

II. RELATED WORK

Several works have been proposed to improve the yield by classifying faults as acceptable faults and unacceptable faults. These employ different techniques such as integer linear programming [6], sampling methods for error estimation [7], threshold based test generation [8] etc. Further, the work [9] shows a technique to generate tests efficiently if such a classification is available.

However, all these approaches are applied to conventional circuits without taking into consideration the errors introduced as part of the design process itself. Therefore, these approaches cannot be directly applied to AC. It has to be noted that “normal” circuits that produce errors due to manufacturing defects do not constitute approximation circuits. In AC, errors are introduced into the design for high speed or low power. In other words the error is already introduced and taken into consideration during design time. Now if for these designed approximated circuits arbitrary fabrication errors are allowed, the error effects will magnify. For instance, if we discard all the stuck-at faults at the lower bit of an approximation adder under a worst-case error constraint of at most 2, the resulting error can in fact increase above the designed limit. Therefore, the AC application will fail under such defects. This is exemplified later in a motivating example in Section IV-A. The key of our
work is that we identify all faults which are guaranteed not to violate the given error metric constraint, coming from the AC application. This ensures that the AC chip will work as originally envisioned for.

At this point we differentiate our work from [10]. In [10] (and closely related [11]), structural analysis is used to determine the most vulnerable circuit elements. Only for those elements test patterns are generated and this approach is called approximate test. In addition, note that [10] targets “regular” non-approximated circuits and therefore we categorize it as a technique for approximating a test, rather than a technique for testing an already approximated circuit.

III. Preliminaries

At first, the relevant parts from post production tests are reviewed in this section. Then, the basic error metrics typically used in approximate computing are defined and it is reviewed how to precisely compute them.

A. Post Production Test, Faults and ATPG

The manufacturing process of a circuit is vulnerable to a large number of physical defects, especially due to the shrinking feature sizes. A post production test is applied to the manufactured ICs to detect these defects and filter out the non-correct circuits. A fault $f$ is a logical manifestation of these defects. The most popular fault model used in practice is the Stuck-At Fault Model (SAFM). In this scheme, a signal connection $s$ in the circuit is considered to be permanently ‘stuck’ at a constant value, either 1 or 0. In this work, we concentrate only on the SAFM, but the proposed approach can also be extended to other fault models.

A test set $T$ is a set of test vectors $t_1, \ldots, t_n$ applied at the circuit inputs which activates the fault locations and produces detectable difference at an observation point. In the post-production test, each detectable fault in the circuit has to be covered by at least one test pattern in the test set. The computation of this test set is called Automated Test Pattern Generation (ATPG) [12]. The ATPG takes all faults $F = f_1, \ldots, f_m$ of a fault model as input and generates a favorably small test set with a high fault coverage.

Basically, a fault $f$ can be classified by the ATPG in three categories. A fault is called detectable, when the ATPG proves that the fault is testable by producing a test which detects $f$. A fault is redundant, when the ATPG proves that there is no test which is able to detect $f$. A fault is classified as aborted, when the ATPG cannot classify $f$ due to reasons of complexity. ATPG techniques are well developed and are able to produce small test sets in reasonable time using structural implication techniques or formal proof engines [13].

B. Error Metrics

Several error metrics have been proposed to determine the quality of approximations (see e.g., [4], [14], [5]). The metrics relevant for this work are given in the following.

Let $N$ be a netlist with $p$ input bits and $q$ output bits, and $N_f$ be the version with an injected fault $f$. The error $e_f$ of the netlist under the fault $f$ is the absolute difference in the magnitudes of the output. i.e., error $e_f$ over $q$ output bits is

$$e_f = |(N_{f,q-1}, \ldots, N_{f,0}) - (N_{q-1}, \ldots, N_0)|$$

Here $(N_{f,q-1}, \ldots, N_{f,0})$ and $(N_{q-1}, \ldots, N_0)$ are the output bit vectors of the faulty and original netlist respectively.

The worst-case error or error-significance is the maximum possible error magnitude among all the $2^p$ input combinations.

$$wc = \max\{e_{f_0}, e_{f_1}, \ldots, e_{f_M}\}$$

where $M = 2^p - 1$.

The error-rate is the ratio of the count of all the errors to the total number of input vectors, i.e.,

$$er = \sum_{i=0}^{M} (e_f \neq 0) / 2^p$$

The total bit-flips $e_f^h$ is the hamming distance between $N$ and $N_f$ due to the injected fault $f$.

$$e_f^h = \sum_{i=0}^{q-1} (N_{f,q-1}, \ldots, N_{f,0}) \oplus (N_{q-1}, \ldots, N_0)$$

And the bit-flip error is the maximum hamming distance possible due to the fault $f$,

$$bf = \max\{e_{f_0}^h, e_{f_1}^h, \ldots, e_{f_M}^h\}$$

All these error metrics are independent quantities on their own and do not necessarily correlate with each other. For instance, a design with very high worst-case error does not imply that the bit-flip error or the error-rate is high.

An approach based on formal methods has been presented in [4] to precisely determine the impact of approximation wrt. a given error metric. The authors propose an approximation miter circuit inspired from the classical formal equivalence checking. In our work we retain the fundamental concepts used in [4], but adapted for fault classification.

IV. APPROXIMATION-AWARE TEST METHODOLOGY

In this section we introduce the proposed approximation-aware test methodology. Before the details are provided, we describe the general idea using a motivating example. In the second half we present the proposed fault classification approach.

A. General Idea and Motivating Example

In the context of approximate computing yield improvement can be achieved when a fault (logical manifestation of a physical defect) is found which can still be tolerated under the given error metric. In this case the fabricated chip can still be used as originally intended, instead of sorting it out. As mentioned earlier we consider single stuck-at faults in this work only (cf. Section III-A). Given an approximate circuit, a constraint wrt. an error metric, the list of all faults for the approximate circuit, then each fault is categorized by our approach into one of the following:

- **approximation-redundant fault** – These are faults which can be approximated, i.e. the fault effect can have an observable effect on the outputs, but it is proven that the effect will always be below the given error limit. Hence, no test pattern is needed for these faults. Note that regular redundant faults are also classified into this category.
- **non-approximation fault** – These are faults whose error behavior is above the given error limit. Hence, they have
to be tested in the post production test and thus a test pattern has to be generated for these faults.

Further, if a fault cannot be classified due to reasons of excessive run time, it is treated as an non-approximation fault.

In the following a motivating example is provided to demonstrate both fault categories. Consider the 2-bit approximation adder as shown in Fig. 1. This adder has two 2-bit inputs $a = a_1 a_0$ and $b = b_1 b_0$ and the carry input $c_{in}$ and computes the sum as $c_{out} = \text{sum}_1 \text{sum}_0$. The (functional) approximation has been performed by cutting the carry from the full adder to the half adder as can be seen in the block diagram on the left of Fig. 1. As error metric we consider a worst-case error of 2 (coming from the application where the adder is used). To explain the proposed fault classification we will focus on the output bit $\text{sum}_0$ and the faults at this bit, i.e. $f_{\text{SA0}}$ and $f_{\text{SA1}}$ corresponding to a stuck-at-0 and stuck-at-1 fault, respectively.

The truth table of the original golden adder, the approximation adder, and the approximation adder with different fault manifestations is given at the right side of Fig. 1. The first column of the truth table is the input applied during fault simulation, followed by the output response of the correct golden adder. Next the response of the approximation adder, and the faults at this bit, i.e. $f_{\text{SA0}}$ and $f_{\text{SA1}}$ corresponding to a stuck-at-0 and stuck-at-1 fault, respectively.

The truth table of the original golden adder, the approximation adder, and the approximation adder with different fault manifestations is given at the right side of Fig. 1. The first column of the truth table is the input applied during fault simulation, followed by the output response of the correct golden adder. Next the response of the approximation adder, and the faults at this bit, i.e. $f_{\text{SA0}}$ and $f_{\text{SA1}}$ corresponding to a stuck-at-0 and stuck-at-1 fault, respectively.

The truth table of the original golden adder, the approximation adder, and the approximation adder with different fault manifestations is given at the right side of Fig. 1. The first column of the truth table is the input applied during fault simulation, followed by the output response of the correct golden adder. The column of the truth table which becomes evident in column $e^\star$ and the shaded rows.

In practice, the employed error criteria follows the requirements of the AC application. Each application will have a different sensitivity on the error metric given in Section III-B. However, if we can identify many approximation-redundant faults, they do not have to be tested since they can be tolerated based on the given error metric constraint.

In the next section we present the proposed fault classification algorithm which can handle the different error metrics.

### B. Approximation-aware Fault Classification

At first, the overall algorithm is presented. Then, the core of the algorithm is detailed.

#### 1) Overall Algorithm

The main part of the proposed approximation-aware fault classification methodology is the fault-preprocessor. It classifies each fault into the above introduced fault categories and is inspired by regular SAT-based ATPG approaches, since these approaches are known to be very effective in proving redundant faults.

### Algorithm 1 Approximation-aware fault classification

1. function \text{APPROX\_PREPROCESS}(faultList faults, error behav. e)
2. \hspace{10pt} N ← get_network()
3. \hspace{10pt} for each $f \in$ faults do
4. \hspace{20pt} if fault_not_processed($f$) then
5. \hspace{30pt} $N_f ←$ get_faulty_network($f$)
6. \hspace{30pt} $E ←$ get_error_computation_netw(metric($e$))
7. \hspace{30pt} $C ←$ negation_off($e$)
8. \hspace{30pt} $\Phi = \text{construct_miter}(N, N_f, E, C)$
9. \hspace{30pt} result = solve($\Phi$)
10. \hspace{20pt} if result = SAT then
11. \hspace{30pt} set $f_{\text{status}} ← \text{NonApproxFault}$
12. \hspace{30pt} else
13. \hspace{40pt} set $f_{\text{status}} ← \text{ApproxFault}$
14. \hspace{30pt} end if
15. \hspace{20pt} end if
16. \hspace{10pt} end for
17. \hspace{10pt} return faults
18. end function

### Notes

- 1 Golden non-approx. approx faulty cut 2-bit adder responses
- 2 Approx adder with $SA_0, SA_1$ at $sum_0$ $h_{\text{SA0}}, h_{\text{SA1}}$ In $C_{\text{out}}$, $h_{\text{SA0}}, h_{\text{SA1}}$ Out $C_{\text{out}}$, $h_{\text{SA0}}, h_{\text{SA1}}$ $\pm$ error in each case, worst-case errors $\text{wc}^\star, \text{wc}^\star$, $\text{wc}^\star, \text{wc}^\star$
approximate adder example we set $wc = 1$. In case of the motivating example this worst-case constraint that the given error metric constraint is violated. For instance, in the motivating example fault equivalence rules and constant propagation for redundancy removal.

In addition to the SAT techniques mentioned above, several structural techniques are also used in conjunction with the SAT solver for efficiency (see Line 15). This includes for example fault equivalence rules and constant propagation for redundancy removal.

Besides, several trivial approximation-redundant/ non-approximation faults can be identified. Such trivial faults are located near the outputs. An example is a fault affecting the MSB output bits that always results in error metric constraint violation. These can be directly deduced as non-approximation faults through path tracing.

In the next section the experimental results are provided.

V. EXPERIMENTAL RESULTS

We have implemented all algorithms in C++. The input to our program is the gate level netlist of the approximated circuit which is normally used for standard ATPG generation. Now, instead of running ATPG, we execute our proposed approximation-aware fault classification approach (cf. Section IV). This filters out the approximation-redundant faults. From there on the standard ATPG flow is taken.

In the following we report results for approximated circuits using worst-case error and bit-flip error constraints. Considering error-rate is left out for future work.\(^1\)

The experimental evaluation of our approach has been done for a wide range of circuits. For the circuits we have determined the respective error metrics using the public available tool from [4]. In this section we first explain the results using the worst-case error as approximation pre-processing criteria. These results are provided in Table I. The experimental evaluation using the bit-flip error metric is separately explained at the end of this section with Table II. Note that a combination of these error metrics can also be provided to the tool. All the experiments have been carried out on a system with 3.0GHz Intel Xeon CPU. Further, the worst-case error and the bit-flip error are the error metrics coming from the application.

A. Results for the Worst-Case Error Metric

All the results for the worst-case error scenario are summarized in Table I. Before we describe the different benchmarks (four different sets in total) and the obtained results we explain the general structure of the table. The first three columns give the circuit details such as the number of primary inputs/outputs and the gate count. This is followed by the fault count without our proposed approach, i.e. this gives the “normal” number of faults for which ATPG is executed. Note that fault-equivalence and fault-dominance are already accounted in these fault counts (column: $f_{\text{orig}}$). The next

Now the complete problem is encoded as a SAT instance and run a SAT solver. If the solver returns satisfiable – so there is at least one input assignment which violates the error metric constraint – we have proven that the fault is a non-approximation fault (Line 11). If the solver returns unsatisfiable, we have proven that the fault is an approximation-redundant fault (Line 13). This fault does not have to be targeted during the regular ATPG stage.

As explained before our methodology uses SAT calls to determine the worst-case error behavior (cf. Section IV). However, for error-rate not only pure SAT-calls are needed, but model counting. Model counting is a higher complexity problem compared to SAT (#P-complete vs NP-complete) [23].
two columns provide the resulting fault count and reduction in faults using our approximation-aware fault classification methodology (columns: $\Delta$ and $\Delta^\%$). The last column denotes the run-time in CPU seconds spent for our developed approach, i.e., only the pre-processing (Algorithm 1).

1) Arithmetic Circuits: The first two sets in Table I consists of commonly used approximation arithmetic circuits. The first set are manually architected approximation adders primarily used in image processing applications [16], [17], [18]. These designs are available in the repository [19]. As evident from the Table I, a significant portion of the faults in all these designs are approximation-redundant. It can also be seen that such architectural schemes show a wide range in approximation-redundant fault count, even in the same category. For example, among the different Almost Correct Adders [16], ACA_I_N16_Q4 has a far higher ratio of approximation faults compared to the scheme ACA_I_N16_Q8 (67\% vs 48\%). The adder GDA_St_N16_M4_P4 [18] has a far higher ratio of approximation faults compared to the scheme ACA_I_N16_Q4 (67\% vs 48\%). The adder GDA_St_N16_M4_P4 [18] has a far higher ratio of approximation faults compared to the scheme ACA_I_N16_Q4 (67\% vs 48\%).

In the second set, other arithmetic circuits such as fast adders, multipliers, multiply accumulate (MAC) etc., are evaluated. These designs are from [20]. The automated approximation scheme (public available on GitHub [15]) has been used to approximate these circuits. Similar to the architecturally approximated designs, the relative mix of approximation-redundant and non-approximation faults in these circuits also vary widely depending on the circuit structure.

2) Other Standard Benchmark Circuits: We also have evaluated our approach on circuits from the ISCAS-85 [22] and EPFL [21] benchmarks to demonstrate its generality. Our methodology is able to classify a high percentage of faults as approximation-redundant to be skipped from ATPG generation, eventually improving the yield. The highest fraction of approximation-redundant faults is obtained in the ISCAS-85 circuit C2670 (above 80\%). However, there is a wide variation in the relative percentage of faults classified as approximation-redundant. This primarily stems from the structure of the circuit, approximation scheme employed and the error tolerance of the AC application.

B. Results for the Bit-Flip Error Metric

We have taken the same set of designs given in Table I for the evaluation of the approximation-aware fault classification methodology under the bit-flip error metric. The results obtained are summarized in Table II. As mentioned before the bit-flip error is the maximum hamming distance of the output bits of the approximated and non-approximated designs, irrespective of the error magnitude. The Table II shows the approximation-aware fault classification results for architecturally approximated adders [16], [17], [18], [19], arithmetic designs [20], standard ISCAS benchmark circuits [22] and EPFL benchmarks [21]. The results in Table II show a different trend compared to the worst-case error results in Table I. In general, the approximation pre-processor has classified a lesser percentage of faults as approximation redundant in the first category of hand crafted approximated adder designs. This has to be expected since each approximation scheme is targeted for a different error criteria, and therefore has a different sensitivity for each
of these error metrics. Furthermore, these two error metrics are not correlated. As an example, a defect affecting only the most significant output bit has the same bit-flip error as that of a defect affecting the least significant output bit of the circuit. However, the worst-case errors for these respective defects are vastly different. We refer to the individual works [16], [17], [18], [19] etc., for a detailed discussion of the error criteria employed in the design of these circuits. Nevertheless, our approximation-aware fault classification tool is able to classify a significant number of faults as approximation redundant in several circuits provided in Table II.

Overall, the results confirm the applicability of our proposed methodology. Note that, in general the run-times for a SAT based ATPG flow depend mainly on the circuit complexity, size and the underlying SAT techniques. Our approach is also influenced by these factors. Therefore, improvements in SAT-based ATPG has a direct impact in our approach. It is also worth mentioning that the approximation-aware fault classification and the subsequent ATPG generation is a one time effort whereas the actual post production test of the circuit is a recurring one. Hence, the additional effort and run-times are easily justified due to high reduction in the fault count that has to be targeted for test generation.

VI. CONCLUSIONS

In this work, we presented an approximation-aware test methodology. First, we proposed a novel fault classification based on the approximation error characteristics. Further, we showed a formal methodology that can map all the faults of an approximation circuit into approximation-redundant and non-approximation faults. The approximation-redundant faults are guaranteed to have effects that are below the error threshold limits of the AC application. Hence, the subsequent ATPG generation has to target only the non-approximation faults and thereby yield can be improved significantly.

Our methodology can be easily integrated into today’s standard test generation flow. Besides, the experimental results on a wide range of circuits confirm the potential and significance of our approach. Substantial reduction in fault count up to 80% is obtained depending on the concrete approximation and the error metric.

REFERENCES