Testing Approximate Digital Circuits: Challenges and Opportunities

Marcello Traiola, Arnaud Virazel, Patrick Girard, Mario Barbareschi, Alberto Bosio

▶ **To cite this version:**

Marcello Traiola, Arnaud Virazel, Patrick Girard, Mario Barbareschi, Alberto Bosio. Testing Approximate Digital Circuits: Challenges and Opportunities. LATS 2018 - 19th IEEE Latin American Test Symposium, Mar 2018, Sao Paulo, Brazil. pp.1-6, 10.1109/LATW.2018.8349681 . lirmm-03033024

HAL Id: lirmm-03033024
https://hal-lirmm.ccsd.cnrs.fr/lirmm-03033024v1

Submitted on 1 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Testing Approximate Digital Circuits: Challenges and Opportunities

Marcello Traiola[1], Arnaud Virazel[1], Patrick Girard[1], Mario Barbareschi[2], Alberto Bosio[1]
[1]LIRMM - University of Montpellier / CNRS - France
[2]DIETI - University of Naples Federico II - Italy

***Abstract*—In the recent years, Approximate Computing (AxC) emerged as a new paradigm for energy efficient design of Integrated Circuits (ICs).**

AxC is based on the intuitive observation that, while performing exact computation requires a high amount of resources, allowing selective approximations or occasional relaxations of the specifications can provide significant gains in energy efficiency and area reduction. During the manufacturing process, physical defects (either random or systematic) can affect the IC and may be the cause of faults leading to observable errors. These errors (due to faults) may worsen the accuracy reduction, already introduced during the functional approximation and possibly lead it to become unacceptable. This paper aims at investigating the challenges and the opportunities related to the test of AxC ICs.

Index terms - test; ATPG; functional approximation; logic synthesis; digital circuits

I. Introduction

In the last years, the literature introduced the Approximate Computing (AxC) paradigm, based on the intuitive observation that rather than a perfect result, inner operations of a computing system can be selectively inaccurate for providing gains in efficiency (i.e., less power consumption, less area, higher manufacturing yield). Surprisingly, introduced approximation does not significantly affect the output quality [1]–[3].

Indeed, the literature also proved that some computing domains are characterized by the so-called application inherent resilience property, that is the ability of applications to output good-enough results despite the fact that some of the inner operations or involved data are inexact [4]. This way, AxC techniques benefit from such a property whenever inaccuracy implies performance gain. Such inaccuracy can involve the software layers, as well as hardware components, including memories [5].

Some AxC techniques have been successfully applied to digital ICs. The first introduced technique is the so-called *over-scaling based approximation*. Basically, the IC is forced to work outside its specified operating conditions [1]. The classical example is the reduction of the supply voltage under the minimum value. This will turn out in energy saving, but it will introduce timing errors. A different technique is the *functional approximation* [1]. It aims at modifying the circuit structure so that an original function F is replaced by the function G, whose implementation leads to area/energy reduction at the cost of reduced accuracy, meaning that some errors can be observed at the outputs of G. The observed errors represent a variation between the output values of F (precise) and G (approximation). The variation is the accuracy loss measured by means of quality metric(s). For instance, we can cite the Error Rate, that is how many times an error is observed at circuit outputs, and the Error Magnitude, measured as the difference between the golden and erroneous outputs, both formally defined in [1]. So far, several approaches have been proposed for functional approximation, either manual or automated [6]–[15]. Unlike the "precise" logic synthesis they require two extra inputs:

- **Error Metric**: the function for measuring the variation between the precise and the approximate output values;
- **Error Threshold**: maximum acceptable error measured by using the given error metric.

During the manufacturing process, physical defects (either random or systematic) can affect the IC and may be the cause of faults leading to observable errors. Unfortunately, these errors (due to faults) may further reduce the accuracy - already reduced as result of the functional approximation - and may affect outputs more than the allowable amount (i.e., the amount of error is greater than the threshold). In this context, the role of testing is to ensure that the maximum error is never greater than the acceptable error threshold fixed by the final user. In other words, among the whole set of possible test vectors (covering all the detectable faults), we have to select the smallest subset that ensures that no fault will lead to an error greater than the acceptable one. Moreover, we do not have to test for faults leading to an error lower than the threshold since it is still acceptable.

To the best of our knowledge, no solutions have been proposed so far to deal with this problem. In this paper, we will investigate the challenges and the opportunities related to the test of AxC ICs. For this purpose, we propose a methodology to automatically generate test vectors to guarantee that the error introduced by manufacturing defects is still acceptable w.r.t. error metric and threshold. We focus on one error metric (i.e., error magnitude) and we apply our approach on a public benchmark suite [16].

The paper is organized as follows. Section II introduces the flow of the proposed approach and provides a simple example. Experimental results are discussed in Section III. Finally, conclusions are given in Section IV.

II. Proposed Approach

As introduced in the previous section, the goal of this paper is to target the generation of a test set for a given approximate

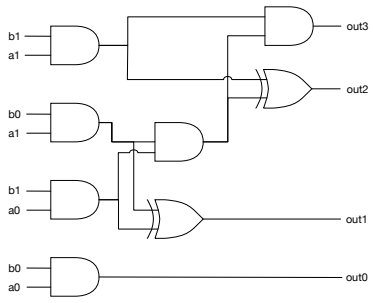
circuit, knowing its error metric and error threshold. In this work, we focus on the Stuck-at Fault model (SaF) and we adopt the Worst Case Error (WCE) as the error metric. WCE is the maximum difference between precise and approximate outputs. It is formally defined in Equation 1.

$$WCE = \max_{\forall i} \left| O^{(i)}_{approx} - O^{(i)}_{prec} \right| \quad (1)$$

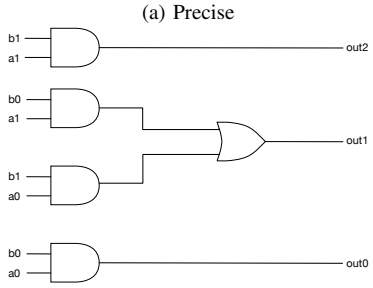
Once the definition of the fault model and the error metric is given, we can exploit a simple example to show the basic principle of the proposed approach and formalize the methodology. This is the purpose of the next subsections.

A. Basic principle

In this subsection, we consider, as a simple example, a two-bit multiplier. The netlist of the precise multiplier circuit and the approximate version are shown in Figure 1a and 1b, respectively. The details about the exploited functional approximation approach are described in [7]. On this simple circuit, the quality metric is the WCE and the error threshold is set to 2. Table I gives all the possible results for the whole inputs space. The only error (highlighted in **red**) appears during the computation of 3 x 3. The result should be 9, but rather we get 7 due to the approximation. However, the erroneous result is still acceptable since the WCE is 2 (9 - 7).



(a) Precise



(b) Approximate

Fig. 1: Two-bit multiplier Example

TABLE I: Error Magnitude Example

A x B	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	4	6
3	0	3	6	**7**

Let us now take into consideration the test of the approximate multiplier of Figure 1b. The classical approach tests for all the possible faults affecting the netlist. More in detail, we use a commercial ATPG [17] to generate the fault list, composed of 44 SaFs, and the test set, composed of 5 test vectors. The achieved fault coverage is 100% (all the 44 faults are tested).

On the other hand, having in mind that the circuit is an approximation of a precise multiplier, we can analyze its outputs to determine what are the faults leading to a non-acceptable error, according to the error threshold. It is worthwhile to emphasize that to perform this analysis we need the knowledge of the error metric, the error threshold and the function implemented by the circuit itself. Otherwise, it is not possible to understand the impact of a fault on the outputs.

TABLE II: Faults Leading to Errors in AxIC multiplier

AxIC fault-free	**AxIC Faulty**		**Diff**
	Sa0@out0	*Sa1@out0*	
0	0	**1**	1
1	**0**	1	1
2	2	**3**	1
3	**2**	3	1
4	4	**5**	1
6	6	**7**	1
7	**6**	7	1

Coming back to our example, let us consider faults affecting *out0*, that is the LSB. The induced error is equal to 2^0 thus lower than the error threshold (WCE = 2). Table II reports all the possible output values obtained by considering the fault-free approximate multiplier, the corresponding outputs considering a SaF at *out0* and the maximum induced error. In other words, for each row, we report the fault-free value (first column), the corresponding values in both cases of Sa0 and Sa1 at *out0* (second and third columns) and the maximum difference between fault-free and faulty values (last column). As expected, such difference is equal to 1, thus we can conclude that faults affecting *out0* should not be tested since the induced error is acceptable (i.e., lower than error threshold).

Unfortunately, the above consideration is not true. Indeed, to determine the real impact of a fault on the outputs of the Approximate Integrated Circuit (AxIC), we must compare the AxIC faulty values with the ones obtained from the precise circuit. Table III reports the output values obtained by considering the fault-free **precise** multiplier (first column) and the faulty AxIC affected by SaF at *out0* (Sa0 and Sa1 shown in the second and third columns). For each output, we highlight in the last column the maximum difference between the fault-free value and the faulty values. In the last row, it is easy to note that the error due to the Sa0@*out0* is non-acceptable. Indeed, the error is equal to 3 and thus greater than the error threshold. Therefore, we have to test for Sa0@*out0* but not for Sa1@*out0* since the latter does not lead to a non-acceptable error.

This simple example has shown that the problem of reducing the test length exploiting the approximation of the circuit under test is not so trivial and needs for further discussion. The

TABLE III: Faults Leading to Errors: Precise VS AxIC

Precise fault-free	AxIC Faulty		Diff
	Sa0@out0	*Sa1@out0*	
0	0	**1**	1
1	**0**	1	1
2	2	**3**	1
3	**2**	3	1
4	4	**5**	1
6	6	**7**	1
9	**6**	7	**3**

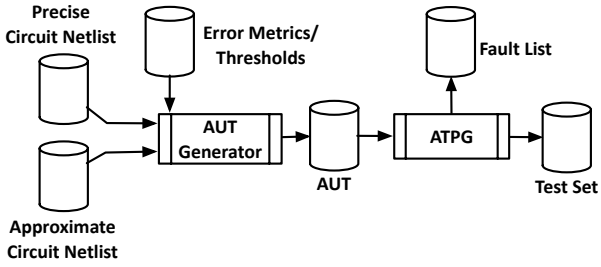


Fig. 2: A schematic view of the proposed flow

next subsection introduces a formalization of the approximate testing procedure.

B. Methodology

Figure 2 sketches the overall flow of the proposed approach. It is composed of two main steps: (i) the *Architecture Under Test* (AUT) Generator and (ii) the ATPG. The AUT generator requires as inputs the approximated circuit netlist, the precise circuit netlist, the error metric and the error threshold.

As pointed out in the previous subsection, the tricky aspect is the comparison of the faulty values w.r.t. the precise values. This is also the main reason why we need the precise circuit netlist. What we propose is to let the ATPG deal with the problem of comparing the outputs of the precise and the approximate circuits. Figure 3 reports a schematic view of an AUT. The basic idea is to create a new circuit that embeds both the precise circuit and the approximate circuit, which take the same inputs (X_1 to X_n in the figure). The outputs are then used to compute the error metric (Error Metric Comp. in the figure). Finally, the computed error *E* is evaluated w.r.t the given error threshold (*Thr* in the figure). If *E* is lower than *Thr*, then the output *G/NG* will be set to logic-1 (Good), otherwise *G/NG* will be set to logic-0 (NoGood).

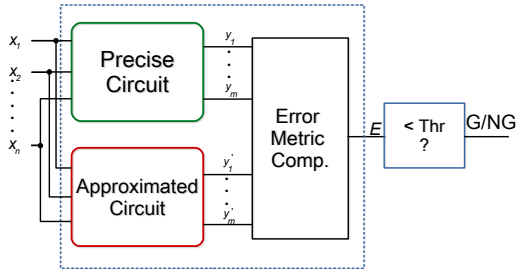


Fig. 3: Architecture Under Test

The only case where a logic-0 may appear at the *G/NG* output is when a fault affecting one of the AUT components leads to an error *E* greater than *Thr*. Thus, during the test generation process, we target only faults affecting the approximate circuit instance. In this way, the ATPG will search for test vectors able to force an error *E* greater than *Thr*. In other words, we test only for the AxIC faults leading to non-acceptable errors. The outputs of the ATPG are the test set and the fault list containing all the faults leading to non-acceptable errors.

In the next subsection, we apply the proposed methodology to the example given in Figure 1.

C. Simple Example

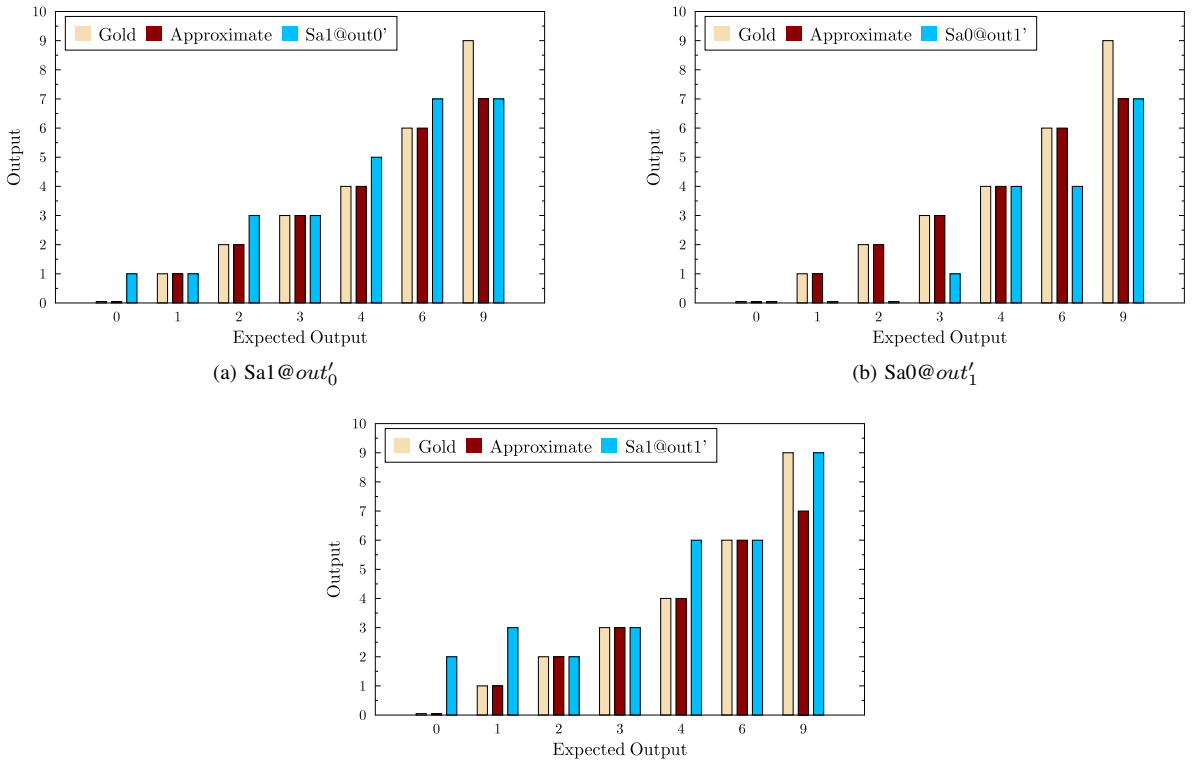
Let us come back to the two-bit multiplier circuit example of Figure 1. By using both precise and approximate circuits, we implemented the AUT as shown in Figure 5. In this specific case, the Error Metric Comp. block implements the absolute difference between the precise and approximate outputs ($|out - out'|$). Since out'_3 has been removed during the approximation, we force it to be set at logic-0 (connected to GND) for the Error Metric Comp.

For this example - and also for the experimental results - we exploited a commercial ATPG [17]. We set the ATPG for considering only the SaFs affecting the approximate circuit (44 SaFs). We ran the ATPG and we obtained 3 test vectors detecting faults leading to having logic-0 at the *G/NG* output. The detected faults were 10 over the initial set of 44. The remaining faults, which led to having logic-1 at the *G/NG*, were classified by the ATPG as Non-Detected. Indeed, they did not lead to any observable output (i.e., the AxIC output error was not greater than the error threshold). We remind that the error threshold is set to 2 in this example.

In order to clarify the underlying idea, Table IV compares the outputs of the precise and approximate circuits (*out* and *out'* respectively) when a Sa0 affects out'_0 (i.e., LSB of the AxIC). Outputs values are represented as binary. In the last row, it is easy to remark that a Sa0 affecting out'_0 leads to a Non-Acceptable value. Indeed, the last row reports the outputs obtained when we provide the input "1111" (i.e., 3 x 3). Even though the precise output is "1001" (i.e., 9), the functional approximated circuit in absence of faults produces "0111" (i.e., 7), which respects the error threshold constraint. The value "0110" (i.e., 6), due to a Sa0 at out'_0, leads to an error greater than the threshold. Thus, it turns to be mandatory to test for that fault. On the other hand, Sa1 affecting out'_0 will not be marked as a fault to test, since for all the cases the difference between *out* and *out'* is lower than 2.

TABLE IV: Sa0 affecting out'_0

out (out_3,out_2,out_1,out_0)	**out'** (out'_3,out'_2,out'_1,out'_0)
0000	0000 → 0000 OK
0001	0001 → 0000 OK
0010	0010 → 0010 OK
0011	0011 → 0010 OK
0100	0100 → 0100 OK
0110	0110 → 0110 OK
1001	**0111** → **0110** KO



(a) Sa1@out'_0

(b) Sa0@out'_1

(c) Sa1@out'_1

Fig. 4: Accuracy Degradation due to Fault Escape

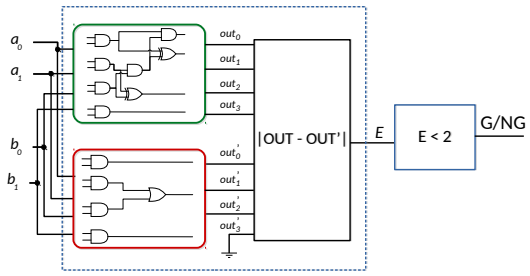


Fig. 5: Two-bit multiplier AUT

To conclude, on this simple example we reduced of about 77% the number of faults (from 44 to 10) and about 40% the test length (from 5 to 3 test vectors). The main advantages of such approach are: (i) the reduce test time and thus test cost, (ii) the fact that less devices will be declared failing leading to eventually increase the yield. Clearly, the good-enough devices (i.e., those for which the manufacturing defects do not lead to unacceptable errors) may have additional accuracy reduction compared to the fault-free approximate circuit, but still acceptable by the given error metric/threshold.

To better clarify this point let us come back to our example. Figure 4 depicts the impact of the untested faults (44 - 10 = 34) on the WCE of the approximate two-bit multiplier. For the sake of simplicity, we report the impact of 3 faults affecting the primary outputs: Sa1@out'_0, Sa0@out'_1 and Sa1@out'_1 on three different histograms (Figure 4a, Figure 4b and Figure 4c). The remaining 31 faults can be considered as equivalent in the sense that their effect impact on out'_0 and out'_1 too.

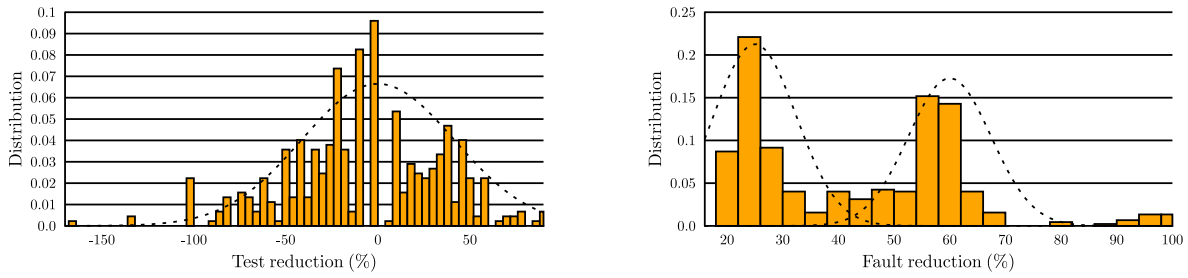
Each histogram reports on the horizontal axis the expected golden output values (decimal) and on the vertical axis the actual output values obtained by the precise multiplier, the fault-free approximate multiplier and the faulty approximate multiplier. The faulty approximate multipliers provide more wrong output values w.r.t. to the fault-free approximate circuit. For example, in Figure 4a we can see that 5 values are wrong while in the fault-free circuit only 1 output value is wrong. In this sense, the accuracy is lower than the fault-free approximate circuit. However, this is not a problem since for the desired error metric (WCE) and error threshold (2) no violations occur and thus the quality of the circuit is the required one.

III. Experimental Results

We validated the proposed methodology by leveraging the public library of approximate components called EvoApprox8b [16]. This library contains 430 non-dominated 8-bit approximate adders (created from 13 conventional adders) and 471 non-dominated 8-bit approximate multipliers (created from 6 conventional multipliers). Both adder and multiplier are crucial components of, for example, approximate image and video processing applications.

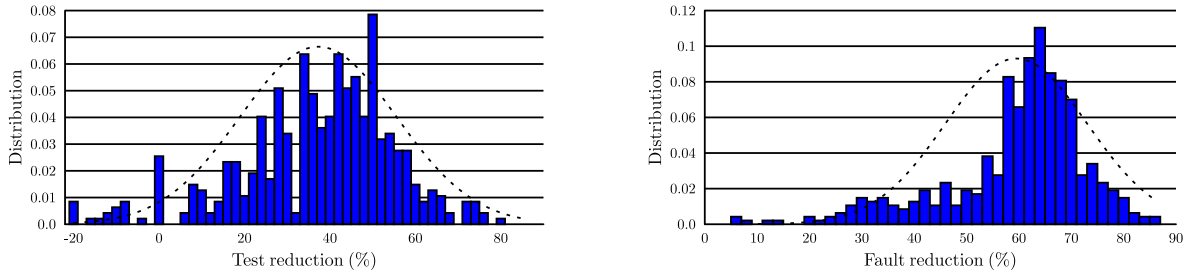
For our experiments, we take into account all the approximated circuits (both adders and multipliers). All the circuits are synthesized using the Cadence Encounter RTL Compiler and TSMC 180 nm library.

As already stated in previous sections, in this work we focus on a single error metric that is the WCE. Table V reports the main statistics on the library adders as well as



(a) Test Reduction (b) Fault Reduction

Fig. 6: Obtained Results for the Approximate Adders



(a) Test Reduction (b) Fault Reduction

Fig. 7: Obtained Results for the Approximate Multipliers

multipliers. We report the Maximum/Minimum number of faults and the Maximum/Minimum WCE. Please note that there is no relation between the two parameters (i.e., the maximum number of faults is not related to the multiplier having the minimum WCE and *vice versa*). As it can be deduced from the number of faults, the adders are smaller (in terms of circuit netlist) than the multipliers.

TABLE V: EvoApprox8b Statistics

	Adders	**Multipliers**
Max #Faults	460	1726
Min #Faults	80	528
Max WCE	64	3204
Min WCE	7	1

The experimental flow that we adopt is the following. First, we run the ATPG for each approximate adder and multiplier in order to test for all the possible detectable faults. We instruments the ATPG using the classical options (static and dynamic compaction) targeting Stuck-at-Faults. This represents the *classical test approach*. Then, for each of them, we apply the proposed methodology. The goal is to show the reduction of both the test length and the number of faults to detect considering the proposed approach against the classical one. Figure 6 and 7 report the obtained results for adders and multipliers respectively. Each figure shows two charts representing the % of Test Reduction (Figure 6a and Figure 7a) and the % of Fault Reduction (Figure 6b and Figure 7b).

For each chart, the horizontal axis plots the % of reduction and the vertical axis the distribution associated with the achieved reduction. This means that for a given reduction (i.e., a given X value) we plot the percentage of circuits achieving that reduction w.r.t. to the total amount of circuits. For example, for the case of the multiplier, the distribution of circuits achieving 65% of fault reduction is 11% (0.11) over the whole set of 471 multipliers (the highest bar in the histogram of Figure 7b). To better discuss the obtained results, we report the main statistics in Table VI. First of all, we can note that for some cases, the number of test vectors increases instead of decreasing as expected. The worst case is reported in Table VI as -166% meaning that we increase the number of vectors of about 166% (for this specific case we increase from 3 test vectors to 8).

To explain the reason behind this result, we can resort to a simple example. Let us consider that 3 faults (f_1, f_2 and f_3) are targeted in the classical approach while only f_2 and f_3 are targeted in our approach. Now, in the former case, it is possible that the test vector targeting f_1 can also detect f_2 and f_3 leading to having only 1 test vector. On the other hand, in the latter case, it is possible that the test vector generated for f_2 does not cover f_3 and thus the ATPG has to generate two test vectors.

As shown in Figure 6 and 7 the negative reduction of test vectors mostly affect the adders (Figure 6), while for the multipliers only a few cases (up to 0.02 %) shown an increment in the number of test vectors. Now the question is: why this difference between adders and multipliers?

To answer this question, we resort to the chart of Figure 8. We plotted, for approximate adders, the % of test reduction against the circuit size (i.e., the number of gates). It is possible to note in the chart a clear zone, from about 60 to 300 gates as circuit size, for which the reduction is negative (i.e., we increase the number of test vector). For larger adders, however, we have a significant amount of test reduction, up to 90 % for the biggest adder (about 500 gates).

Moreover, looking at the multipliers, we obtained a signifi-

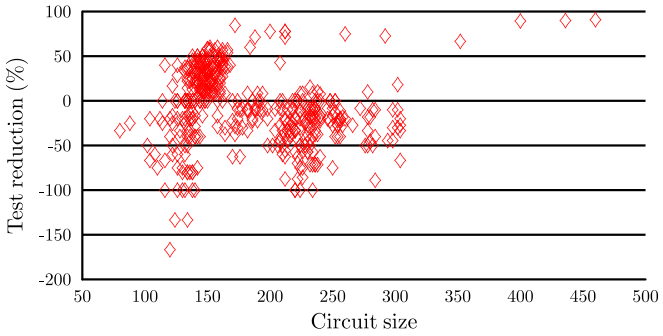


Fig. 8: Adders Size VS Test Reduction

cant test reduction. The maximum value is 79% and in average 37%. Interestingly, the smallest multipliers have a size of about 500 gates that corresponds to the biggest adders. It seems that the proposed approach works well if applied to large circuits.

Please note that for these specific cases, even if we do not have a test time reduction we can still increase the yield because we do not test for acceptable faults (i.e., those leading to acceptable errors). It will depend on the user constraints to chose a tradeoff between test time and yield. However, we plan to solve the issue also for the case of smaller circuits. Indeed, this problem is related to the ATPG algorithm. Thus a possible solution could be not to generate a new test set (as we did in this work), but reducing the test set provided by the classical test approach (the one covering for all detectable faults) exploiting unspecified bits.

We had a similar result for the faults reduction. As for the multipliers, we obtained a reduction from 5% up to 85% (in average 59%); as for the adders, from 18% up to 99% (in average 42%). Clearly, the number of faults cannot be increased since we do not modify the circuit structure during the AUT generation.

TABLE VI: Reduction Statistics

		Adders	**Multipliers**
Test Reduction	avg	-6%	37%
	max	90%	79%
	min	-166%	-33%
Fault Reduction	avg	42	59%
	max	99	85%
	min	18	5%

IV. Conclusions

In this paper, we presented the problems related to the test of approximate digital circuits. To the best of our knowledge, this is the first paper targeting testing of approximate digital circuits. In this context, the role of testing is to ensure that the maximum error is not greater than the acceptable error threshold. In other words, among the whole set of possible test vectors (covering all the detectable faults), we have to select a subset that ensures that no fault will lead to an error greater than the acceptable one. The proposed approach for generating test vectors targeting approximate ICs has been validated on a publicly available benchmark library. The paper shown that there are opportunities in terms of test length reduction for AxIC. However, the challenge is to solve the issue of generating extra test vectors, as the case of adders. Moreover, we have to explore the case where more than one error metric is considered: how to model the problem to make possible the use of classical ATPG tools? Finally, we have to deal with more complex circuits for which the error metric computation may be complicated (e.g., the PSNR).

References

[1] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, Mar. 2016. [Online]. Available: http://doi.acm.org/10.1145/2893356

[2] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, Feb 2016.

[3] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 113.

[5] V. K. Chippa, S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing: An integrated hardware approach," in *Asilomar Conference on Signals, Systems and Computers*. IEEE, 2013, pp. 111–117.

[6] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Transactions on Computers*, vol. 64, no. 4, pp. 984–994, April 2015.

[7] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *2011 24th Internatioal Conference on VLSI Design*, Jan 2011, pp. 346–351.

[8] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, March 2010, pp. 957–960.

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

[10] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 1367–1372.

[11] J. Miao, A. Gerstlauer, and M. Orshansky, "Multi-level approximate logic synthesis under general error constraints," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2014, pp. 504–510.

[12] Y. Wu and W. Qian, "An efficient method for multi-level approximate logic synthesis under error rate constraint," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2016, pp. 1–6.

[13] D. Shin and S. K. Gupta, "A new circuit simplification method for error tolerant applications," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[14] A. Ranjan, A. Raha, S. Venkataramani, K. Roy, and A. Raghunathan, "Aslan: Synthesis of approximate sequential circuits," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2014.

[15] L. Holik, O. Lengal, A. Rogalewicz, L. Sekanina, Z. Vasicek, and T. Vojnar, "Towards formal relaxed equivalence checking in approximate computing methodology," *2nd Workshop On Approximate Computing (WAPCO)*, 2016. [Online]. Available: https://wapco.e-ce.uth.gr/2016/papers/SESSION2/wapco2016_2_1.pdf

[16] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapproxsb: Library of approx adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 258–261.

[17] (2017) Tetramax. [Online]. Available: https://www.synopsys.com/