EINDHOVEN UNIVERSITY OF TECHNOLOGY

Embedded toggle generator to control the switching activity during test of digital 2D-SoCs and 3D-SICs

Citation for published version (APA):
Katselas, L., Athanasiadis, A., Hatzopoulos, A., Jiao, H., Papameletis, C., & Marinissen, E. J. (2017). Embedded toggle generator to control the switching activity during test of digital 2D-SoCs and 3D-SICs. In *Proceedings of IEEE International Symposium on Power and Timing Modeling, Optimization, and Simulation 2017* (pp. 1-8). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/PATMOS.2017.8106969

DOI:
10.1109/PATMOS.2017.8106969

Document status and date:
Published: 27/09/2017

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 16. Nov. 2023

2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)

Embedded Toggle Generator to Control the Switching Activity during Test of Digital 2D-SoCs and 3D-SICs

Leonidas Katselas[1] Angelos Athanasiadis[1] Alkis Hatzopoulos[1]
Hailong Jiao[4*] Christos Papameletis[3] Erik Jan Marinissen[2,4]

[1] Aristotle Univ. of Thessaloniki
University Campus
54124 Thessaloniki
Greece
katselas@auth.gr
angeatha@ece.auth.gr
alkis@eng.auth.gr

[2] IMEC
Kapeldreef 75
B-3001 Leuven
Belgium
erik.jan.marinissen@imec.be

[3] Cadence Design Systems
1701 North Street
Endicott, NY 13760
United States of America
christos@cadence.com

[4] Eindhoven Univ. of Technology
Den Dolech 2
5612 AZ Eindhoven
The Netherlands
h.jiao@tue.nl

***Abstract*—In digital logic circuits, unconstrained scan tests are known to evoke much higher switching activity than functional modes. To create test conditions which are as similar as possible to functional modes, today's ATPG tools have knobs to constrain the switching activity of the generated test to a user-defined functional (= lower) level. Two-dimensional system chips (SoCs) and three-dimensional stacked ICs (SICs) are typically tested in a modular fashion, i.e., per embedded core or stacked die. At any moment during the test, one or more modules are being tested ('module-under-test', MUT); we refer to the modules currently *not* being tested as 'neighbors'. The switching activity of the MUT(s) can be controlled by ATPG constraints, but the switching activity of the neighboring modules is typically not controlled. In this work, we present two key elements for an approach to control the switching activity of both MUT(s) and neighboring modules. The first is a toggle analysis tool, that determines the switching activity of a module in either functional or test mode on the basis of a Value Change Dump (.vcd) file generated during gate-level Verilog netlist simulation. The second element is a programmable on-chip toggle generator, for which we present both its hardware scheme, as well as an algorithm to program it to achieve any target switching activity. For each module, the toggle analysis tool can be used to determine the switching activity in functional mode, which then forms the target for the ATPG tool when the module is a MUT, or for its embedded toggle generator while the module is in its role as neighbor.**

I. Introduction

Today integrated circuits (ICs) have evolved into very complex system chips. Their designs consist of many hierarchical levels, including cores, subsystems and (in the case of 3D-SICs) dies. Testing these large chips for manufacturing defects in a modular fashion has major benefits with respect to test quality, test development effort, and test application cost[1]. Modular test performs test generation per module, which makes ATPG and fault-simulation much more tractable tasks. Such a divide-and-conquer test approach minimizes the required test data volume[2] and offers opportunities to reduce test application times through effective scheduling [3]. In modular test, a module alternatingly assumes the roles of module-under-test (MUT) and module-not-under-test (which we refer to as 'neighbor' to the MUT). Modular testing typically involves wrapping test modules for controllability and observability purposes and connecting all test wrappers to SoC pins through one or more test access mechanisms (TAMs). Test wrappers have been standardized by IEEE Std 1500 [1, 4, 5].

Although leakage power is increasingly important for advanced CMOS technology nodes, the dominant power consumption component remains dynamic power; it is linearly proportional to the amount of switching activity in the circuit. In high-performance applications, massive switching might lead to faults due to critical IR-drop in the power distribution network and (over-)heating. In mobile low-power applications, excessive switching reduces battery life. Hence, designers have come up with various ways to reduce the amount of switching in the functional operation of the system, such as clock gating, multiple clock domains, dynamic frequency scaling, and multiple power domains [6, 7].

During test, the switching activity is typically significantly higher than during functional operation. Peak power increases of up to 30$\times$ during scan test as compared to the functional mode have been reported [8]. During manufacturing test, power is typically provided from a wall-plugged ATE, and hence battery life is not a concern; however heat dissipation, excessive IR drop, and providing realistic test conditions that mimic functional operation are. Consequently, EDA suppliers have equipped their ATPG tools with low-power modes and pattern compression techniques, in which the switching activity can be constrained by the user, often at the expense of additional test patterns and/or a slight drop in fault coverage [9–13].

In many test access architectures, the neighbors are toggling on the basis of the MUTs test pattern data while shifted in

*Hailong Jiao is currently affiliated to Peking University Shenzhen Graduate School in Shenzhen, China. Reach him at jiaohl@pkusz.edu.cn.

978-1-5090-6462-5/17/$31.00 ©2017 IEEE

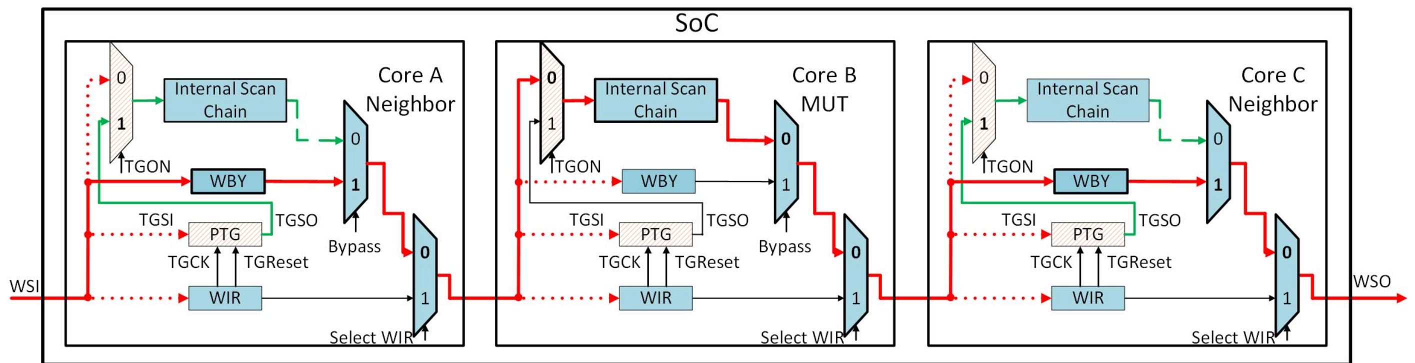

Fig. 1: An example of a SoC during modular test. Core *B* is the MUT, and the Core *A* and the Core *C* are neighbors.

and out of the SoC/SIC [14, 15]. The test patterns targeted at the MUT are random data to the neighbors, and hence evoke arbitrary, typically high, switching activity in these neighbors. This might cause IR drop, over-heating, and/or cross-talk noise. Consequently, the test conditions for the MUT(s) might be unnecessarily harsh, due to which a MUT might fail its test unjustifiably ('false reject') or it might be permanently damaged due to excessive switching activity. In both cases this results in unnecessary yield loss.

Some SoC/SIC designs prevent random toggling of neighbors by shielding them from the MUT test pattern data that is passing by, e.g. by gating the internal scan chain(s), clock gating, and/or power-down modes [16]. This easy-to-implement approach reduces the switching activity in neighboring modules to zero. Unfortunately, it might provide a too-optimistic test condition for the MUT(s) and consequently lead to test escapes ('false positives') [17]. False rejects lead to unnecessary yield loss, which is typically expressed as a percentage. On the other hand, test escapes are typically measured as defective parts-per-million (DPPM); just the relative size of these two units is an indication that test escapes are even more intolerable than yield loss due to false positives.

In this work, we aim to provide realistic test conditions to the MUT(s), to minimize both yield loss and test escapes [17]. Therefore, we want to control the switching activity of modules during test, also when these modules are in their neighbor role. To this end, we equip all modules with an on-chip programmable toggle generator. The toggle generator can be programmed to impart a toggle pattern to the module-internal scan chains that causes a user-defined switching activity within the module when in its neighbor role. Typically, we set the user-programmable switching activity to be equal or similar to the switching activity of the functional mode of that module.

Figure 1 illustrates the concept. It shows an SoC consisting of three cores: *A*, *B*, and *C*. All three cores have an IEEE 1500-compliant test access structure (shown in blue) consisting of a Wrapper Instruction Register (WIR), a Wrapper Bypass (WBY), core-internal scan chains, and some mode-configuring multiplexers. Shown in light-shaded orange are the DfT extensions proper to our approach: a Programmable Toggle Generator (PTG) and an additional multiplexer. In the configuration shown in Figure 1, Core *B* is in its MUT role, while Cores *A* and *C* are in their neighbor roles. The bold red solid line shows the test data flow: test stimuli from the test equipment, via the Wrapper Serial In (WSI) pin, bypassing Core *A* through its WBY register, into the internal scan chains of Core *B*; and the test responses from the internal scan chains of Core *B*, bypassing Core *C* through its WBY register, via the Wrapper Serial Out (WSO) pin, into the test equipment. If the orange-shaded extensions were not present, the neighbors of Cores *B*, Cores *A* and *C* would be toggling at random on the bypassing test data for Core *B*. However, with these extensions, their core-internal scan chains are loaded with scan-test stimuli that are generated by their respective on-chip PTGs. These PTGs are programmed such that a switching activity equal or similar to a functional mode of operation is achieved. While Figure 1 shows a one-bit ('serial') test interface, the same concept is applicable for a multi-bit test interface and parallel scan chains.

This paper presents two key elements for the concept described above with Figure 1. The first element is a toggle analysis tool that can determine the switching activity of a design module in either functional or test mode. The analysis is based on a Value Change Dump (.vcd) file generated during gate-level IEEE Std 1364 Verilog [18] netlist simulation. The second element is the programmable toggle generator (PTG) that is to be included on every design module. We present its hardware scheme, as well as an algorithm to program it to achieve virtually any target switching activity. For each module, the toggle analysis tool can be used to determine the switching activity in functional mode, which then forms the target for the ATPG tool when the module is a MUT, or for its embedded toggle generator while the module is in its neighbor role..

The rest of the paper is organized as follows. Section II describes our toggle analysis tool. The toggle generator is described in Section III. Experimental results based on ISCAS'89 benchmarks and an actual industrial circuit are shown in Section IV.We conclude the paper in Section V.

II. Toggle Analysis Tool

Several industrial EDA tools provide a detailed power analysis of digital ICs. However, the switching activity per net or per clock cycle of a circuit is not available in the power reports generated by those EDA tools. This information is critical to the creation of a system to control the toggle rate of a digital circuit.

As input for the toggle analysis tool we use the Value Change Dump (VCD) file from the functional or test simulation of the circuit-under-analysis. The VCD file format is part of the IEEE Std 1364, the standard for hardware description language Verilog [18]. VCD files contain time-stamped information about value changes (net toggles) on selected nets in the design. Designers can indicate which variables (nets) should be 'dumped' in a Verilog testbench file.

The toggle analysis tool can extract two kinds of circuit profiles on functional or scan test mode. The first profile is the *time profile* (TP) of a circuit. A TP shows the number of net toggles per clock cycle. The second profile is the *net profile* (NP) of a circuit. The net profile (NP) shows the number of toggles per net.

In order to illustrate the capabilities of our toggle analysis tool, we present its results for an industrial IC design D1. Figure 2 shows the TPs of four functional modes of D1. The total switching activity and the standard deviation of net toggles per clock cycle are different for every functional mode. The time profiles for standard and low-power ATPG test are plotted in Figure 3. For reasons specific to the design of D1, the test patterns are scanned in-and-out of the circuit without scan overlap and cause the v-shaped time profile. Consequently, the number of net toggles is increasing during scan load and is decreasing during scan unload. The peaks of the profile occur during the capture phase. Note that the number of net toggles for both test modes is significantly higher (up to 50×) than for any of the four functional modes. Also note that this simulation data shows that the number of net toggles of low-power ATPG is significantly lower than for standard ATPG, confirming the effectiveness of low-power ATPG. Figure 4 shows which fraction of the nets are actually toggling during functional and test modes. This data shows that for the four functional tests the fraction of toggling nets is rather small, while both test modes toggle virtually all nets (as can be expected from a high-quality test). Some nets toggle twice during every clock cycle: they are the so-called 'clock nets'.

A key metric that can be extracted from the proposed analysis is the switching activity (SA). The global Switching Activity summarizes all of the switching of a circuit in a single number. The average percentage of nets that toggle per clock cycle is calculated as follows:

$$SA = (CC \times N)^{-1} \cdot \sum_{i=1}^{N} \sum_{j=1}^{CC} t_{ij}, \tag{1}$$

where CC is the total number of clock cycles. N is the number of nets of the circuit. If net i toggles in clock cycle j, then t_{ij} is 1, and 0 otherwise.

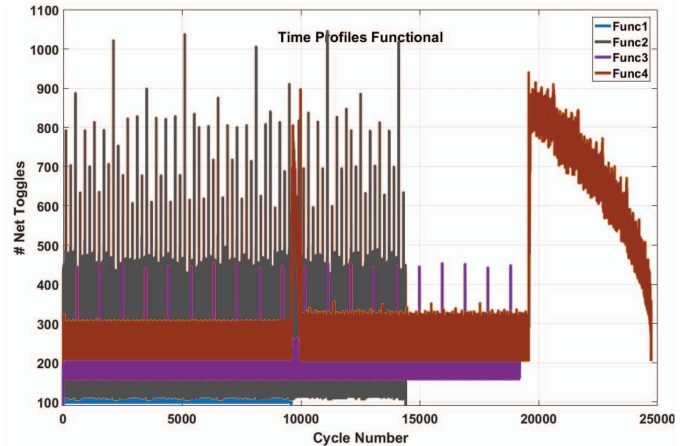

Fig. 2: Time profiles of four functional modes for D1.

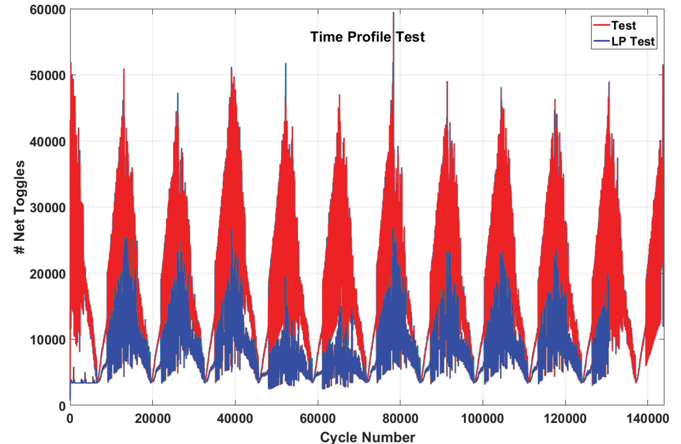

Fig. 3: The time profile of ten standard and low-power test patterns for D1.

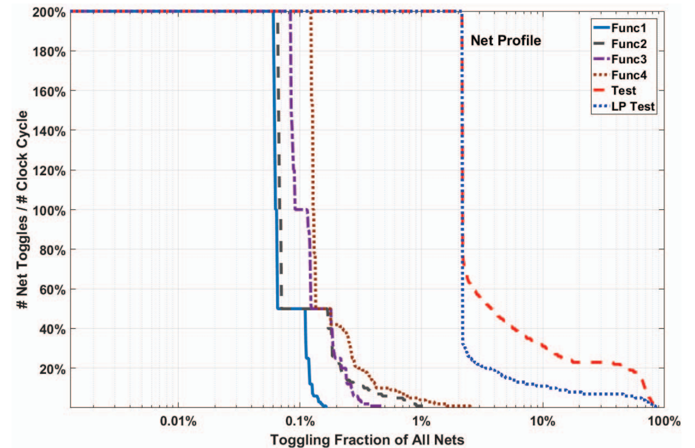

Fig. 4: Net profiles of functional and test modes for D1.

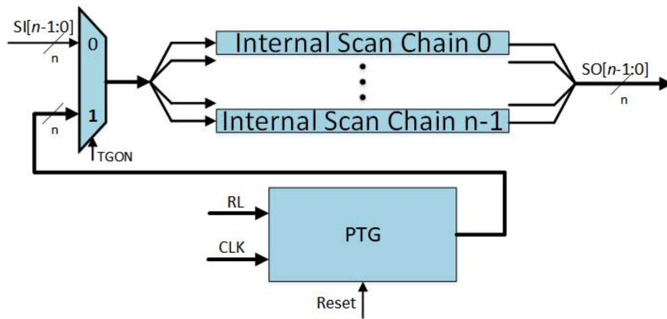

Fig. 5: Toggle generator connected to multiple scan chains.

Dividing the total toggle count by the number of nets and the number of clock cycles (as per Equation (1)) allows to compare the time profiles of different circuits and/or different simulation lengths, such as functional vs. test simulations.

Our toggle analysis tool can be used to analyze the switching activity for any functional or test-mode simulation. A potential issue is that, for a large circuit and a simulation spanning many clock cycles, the simulation time required to create the VCD dump file and the size of the resulting VCD file might grow unwieldly large. While such a large VCD file is no problem for our toggle analysis tool, it could cause storage and transfer issues. Especially scan-test modes are time-consuming to simulate, due to the relatively long scan-in/out times. These challenges can be effectively addressed by only simulating a small subset of all test patterns.

III. Toggle Generator

The hardware scheme of the programmable toggle generator and the algorithms to program the operation of the toggle generator are presented in this section. The toggle generator is controlled by a serial input port and generates a repeated 'toggle pattern'. The toggle pattern is loaded parallel through the scan chains and determines the toggle rate of the circuit. Figure 5 shows the connection of the toggle generator with the scan chains of the circuit. A toggle pattern consists of an endlessly repeated sequence of alternating runs of 0s and runs of 1s of equal run length.

The operation of the toggle generator is separated in two phases, the store phase and the toggle phase. The store phase starts at the beginning of the modular testing. During store phase a serial vector is stored in the register of the toggle generator. This vector contains the information about the *run length* (RL) of the toggle pattern. After completion of store phase, starts the toggle phase. During this phase the toggle pattern is generated and repeated until the end of the modular testing. To achieve small toggle rate, the run length of the toggle pattern be long.

A. Hardware Scheme

The hardware implementation of the programmable toggle generator with a single toggle pattern is shown in Fig 6. It consists of a serial in - parallel out shift n-bit Run Length Register (RL-Register), a digital counter, and a comparator.

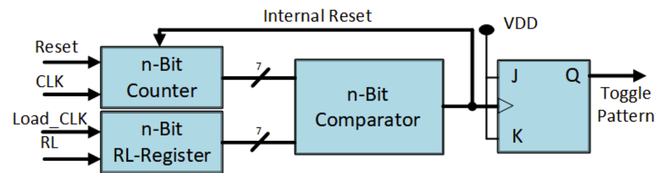

Fig. 6: The hardware scheme of the single toggle generator.

The run length of the toggle pattern (2^n) is loaded and stored as a vector in the register during the store phase. When the output of the counter is equal to the value that is stored in the register, the output of comparator triggers the flip-flop and the output bit changes from 0 to 1 or vice versa. The counter has two reset signals, an internal and an external. The internal reset signal is activated when the comparator is high. The external reset is activated at the start of the modular test.

The toggle generator that combines two toggle patterns is shown in Fig. 7. During the load phase a vector is stored in a N-bit register. The vector is composed of four parts. The first 2n bits indicate the run length of the two toggle patterns. The last 2m bits define how many times a toggle pattern should be repeated to generate the final combined toggle pattern. For example the vector 0011-0010-0010100-0001110 composes a 14-bit long toggle pattern repeated two times and a 20-bit long a toggle pattern repeated three times. During the toggle phase, the m-bit counter and comparator are responsible to repeat the toggle patterns and switch the output of the multiplexers. This toggle generator occupy larger area than the previous, however, the achieved switching activities are closer to the desired values.

B. Single Toggle Pattern Algorithm

The pseudocode of the proposed algorithm to generate toggle patterns with a single toggle pattern is presented in Algorithm 1. Firstly, the target switching activity (TSA) and the target standard deviation (TStdev) are read. The desired delta, the absolute difference of achieved SA and standard deviation from the desire metric, are also defined. Then the user defines the values of the minimum and the maximum run length of the toggle pattern as well as the iteration step from the minimum to the maximum run length.

Algorithm 1 [Single Toggle Pattern]

```
1:  read TSAfunc, TStdDev, DeltaSA, DeltaStdDev;
2:  set min_RL, max_RL, step;
3:  for i = min_RL to max_RL : step;
4:      simulate pattern_i;
5:      calculate SA_i, StdDev_i;
6:  end
7:  for i = min_RL to max_RL : step;
8:      if |SA_i − TSA|<DeltaSA AND |TStdDev_i − TStdDev|<DeltaStdDev
9:          togglePattern = pattern_i;
10:     end
11: end
```

All the toggle patterns are simulated with the target circuit. The SA of each circuit is calculated by using our toggle analysis

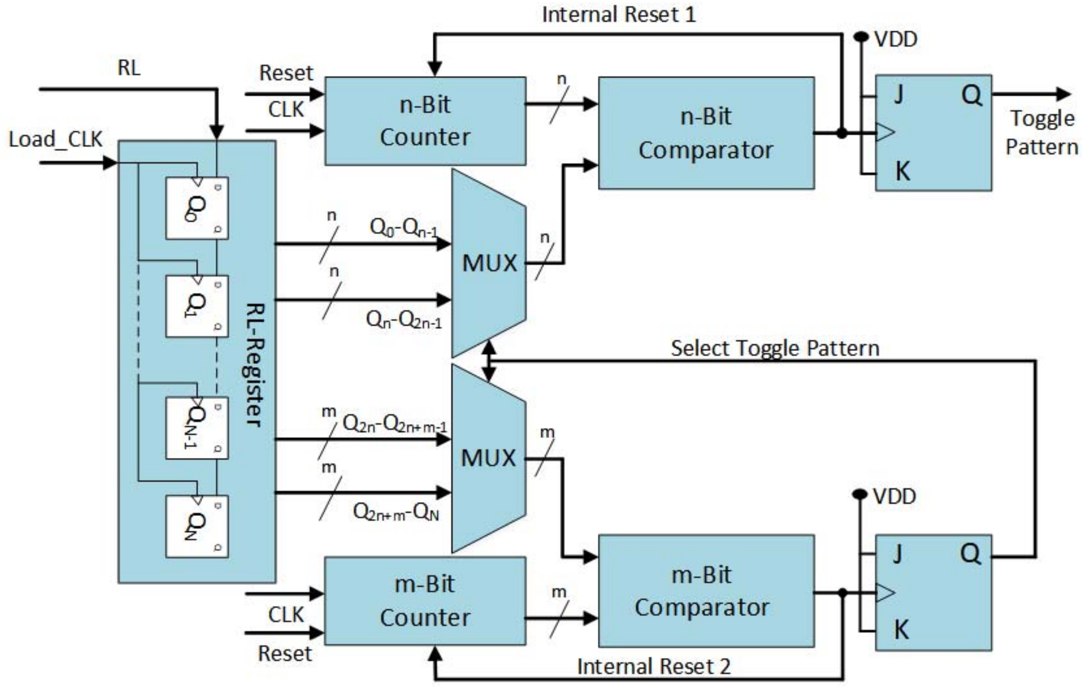

Fig. 7: Hardware scheme of combine toggle generator.

tool. If the SA of a simulation meets the criteria, then this pattern is defined as the toggle pattern. If none of the patterns can be used as toggle pattern, then the user have to change one or more of the parameters to generate the required toggle pattern or apply alternative algorithm that is introduced in the next sub-section.

C. Combined Toggle Pattern Algorithm

If the user demands more accurate SA for the circuit or fewer iterations to get the required toggle patterns, then two different patterns can be combined. The pseudocode of this algorithm is presented in Algorithm 2.

Two adjacent pattrens are selected from the toggle pattern table. The TSA should be between the SA's of the two selected patterns. To generate an accurate toggle pattern, the SA's of the two patterns should be the next higher and smaller ones, respectively. For the sake of convenience, the pattern with the lower SA is called *patternL* and *patternH* the other.

Algorithm 2 [Combined Toggle Pattern]

1: **for** i = *min_RL* **to** *max_RL* : *step*;
2: **if** SA_i >*TSA* **AND** SA_{i+step} <*TSA*;
3: **break**;
4: **end**
5: **end**
6: *interpolation* = $|SA_i - TSA| / |TSA - SA_{i+step}|$;
7: **find** integers A, B **that** A/B $\simeq$ *interpolation*;
8: *togglePattern* = **repeat**(patternL$_i$,B**repeat**(patternL$_{i+step}$,A);
9: **end**

To get as close as possible to the target switching activity, the interpolation of patternL and patternH is used. The absolute difference of the SA$_{patternL}$ and TSA is divided by the absolute difference of the SA$_{patternH}$ and TSA (Line 6). Then we choose the minimum integers (A & B) whose fraction (A/B) is closer to the result of the division (Line 7). The final toggle pattern is the *patternL* repeated B times followed by the *patternH* repeated A times.

D. Silent Scan Chain Algorithm

In cases when the length of the scan chains is shorter than the length of the minimum toggle pattern, the SA of the toggled core is dramatically decreased for some time period because of the lack of toggles. To prevent the long low or high toggle rate periods and sustain a low the standard deviation of the profile, the Silent Scan Chain algorithm is applied.

Algorithm 3 [Silence Scan Chains]

1: **read** *num_sc*;
2: **for** j = 1 **to** *num_sc*
3: **simulate** *min_RL*;
4: **calculate** *scSA*[*j*];
5: **end**;
6: **for** i = *min_RL* **to** *max_RL* : *step*;
7: *num_silen_sc* = (*TSA*/SA_i) $\times$ *num_sc*;
8: **choose** *num_silen_sc* scan chains with the lowest SA from *scSA*[*j*];
9: **simulate** *pattern*$_i$ **on** *num_silen_sc*;
10: **calculate** SA_i;
11: **if** $|SA_i - TSA|$<*DeltaSA* **AND** $|TStdDev_i - TStdDev|$<*DeltaStdDev*
12: **break**;
13: **end**
14: **end**

In Algorithm 3, firstly, the number of the scan chains in the circuit is read. Then we simulate the desired toggle pattern

TABLE I: Design and simulation data.

Circuit Name	#Nets	#Scan Chains	FF/SC
s5378	4,128	2	89
s9234	2,843	2	105
s15850	9,045	3	178
s38417	25,665	3	545
s38584	29,756	3	475
D2	31,521	3	343

TABLE II: Toggle pattern list.

Run Length	s5378	s9234	s15850	s38417	s38584	D2
2	83.08%	49.80%	48.14%	60.15%	63.86%	59.92%
4	41.89%	25.47%	25.85%	30.50%	33.88%	29.10%
8	21.30%	12.78%	13.85%	16.34%	17.54%	15.50%
16	10.98%	6.24%	7.01%	7.90%	9.00%	7.85%
32	5.38%	3.10%	3.64%	4.39%	4.54%	4.10%
64	2.75%	1.58%	1.86%	2.15%	2.38%	2.08%
126	1.38%	0.80%	0.96%	1.08%	1.20%	1.01%
188	0.95%	0.55%	0.63%	0.81%	0.82%	0.85%
250	0.72%	0.43%	0.51%	0.61%	0.62%	0.65%

on each scan chain to calculate the switching activity. The number of the silent scan chains is calculated. After that, we choose the scan chains with the lowest switching activity and we simulate the toggle pattern only on them. If the switching activity and the standard deviation are within the desired range, we use this toggle pattern. If not, we repeat this procedure for every toggle pattern from the minimum to the maximum run length with the step given.

IV. Experimental Results

The objective of our experiments is to show the effectiveness of our toggle generator in achieving the final switching activity and adaptability in complex circuits.

Five benchmark circuits are used for simulations: five ISCAS'89 benchmarks s5378, s9234, s15850, s38417, s38584 [19] and one industrial design D2. For the ISCAS'89 circuits, speculative toggle rates are used since no functional testbench is available for those circuits. The industrial design is larger and its functional SA is known. The smaller designs (s5378, s9234) have two scan chains and the bigger three. The generated toggle pattern are loaded through the scan chains. Table I shows the number of nets, the number, and the length of the scan chains of the designs

The list of the SA of the patterns is shown in Table II. The switching activity for every circuit that is caused from a toggle pattern is shown in the corresponding cell of the table. The minimum run length is two and the maximum is 250. The step that is used is not constant. At the beginning we double the length until the length is 32. Afterwards, 62 is used as the step, because the changing rate of SA is slow, as the run length is increasing.

To demonstrate the effectiveness of our toggle generator we set six switching activities as target for every circuit. The toggle patterns that are used to produce the target switching activity TSA and Delta are shown in Table III. The run length, the achieved SA, and the difference between the achieved and the target SA are also presented. Algorithm 1 is accurate when long toggle patterns are used. However, the efficiency is decreased as the run length is increased. A method to increase the accuracy even for high TSA is to decrease the step for the generation of the toggle patterns, where more simulations are required.

The data obtained with Algorithm 2 are presented in Table IV. To achieve the selected TSAs, two continuous toggle patterns, the SAs of which are on both sides of the TSA, are combined. The Combined Toggle Patterns Algorithm generates more accurate switching activities than the Single Toggle Pattern Algorithm. The difference between the achieved SA and the TSA is lower than 5% in almost every case. The drawback of the Combined Toggle Pattern toggle generator is that the hardware scheme is more complicated than the Single Toggle Pattern toggle generator, yet it is significantly smaller in comparison to an industrial circuit.

Another factor to be taken into account is the occupied area of the toggle generator on the SOC. Table V shows the size of the core, including the test wrapper and the toggle generator. An advantage of our toggle generator is that achieve accurate switching activities and occupy only a small percentage of the total core area.

V. Conclusion

Accurate toggle rate analysis and toggle generation is critical to low power modular test in 2D-SoCs and 3D-SICs. Accurate switching activity profiles (per time and per net) can be created through the analysis of the VCD files that are extracted from functional, as well as from scan test simulation of digital circuits. Algorithms and hardware schemes are proposed in this paper to generate patterns to control the switching activity of the cores neighboring to the module under test in a modular test scenario. Experimental results on ISCAS'89 benchmarks and an industrial circuit demonstrate that the proposed toggle generators control accurately all the circuits for various switching activities.

Acknowledgements

We thank Carolina Mora Lopez, Steven Redant, Geert Vanwijnsberghe, and Jan-Willem Weijers of IMEC IC-Link for providing us with the design and simulation data of D1.

References

[1] E.J. Marinissen and Y. Zorian, "IEEE Std 1500 enables modular SoC testing," *IEEE Design & Test of Computers*, vol. 26, no. 1, pp. 8–17, 2009, doi:10.1109/MDT.2009.12.

[2] O. Sinanoglu *et al.*, "Test data volume comparison: Monolithic vs. modular SoC testing," *IEEE Design & Test of Computers*, vol. 26, no. 3, 2009, doi:10.1109/MDT.2009.65.

[3] U. Ingelsson *et al.*, "Abort-on-fail test scheduling for Modular SOCs without and with Preemption," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3335–3347, 2015, doi:10.1109/TC.2015.2409840.

TABLE III: Toggle generation for various benchmark circuits with single toggle pattern.

TSA	s5378			s9234			s15850		
	Run length	SA	Delta	Run length	SA	Delta	Run length	SA	Delta
20%	8	21.30%	6.50%	4	25.47%	27.35%	4	25.85%	29.25%
10%	16	10.98%	9.80%	8	12.78%	27.80%	16	7.01%	29.90%
5%	32	5.38%	7.60%	16	6.24%	24.80%	32	3.64%	27.20%
2%	126	1.38%	31.00%	64	1.58%	21.00%	64	1.86%	7.00%
1%	188	0.95%	5.00%	126	0.80%	20.00%	126	0.96%	4.00%
0.7%	250	0.72%	2.85%	126	0.80%	14.28%	188	0.63%	10.00%

TSA	s38417			s38584			D2		
	Run length	SA	Delta	Run length	SA	Delta	Run length	SA	Delta
20%	8	16.34%	18.30%	8	17.54%	12.30%	8	15.50%	22.50%
10%	16	7.90%	21.00%	16	9.00%	10.00%	16	7.85%	21.50%
5%	32	4.39%	12.20%	32	4.54%	9.20%	32	4.10%	18.00%
2%	64	2.15%	7.50%	64	2.38%	19.00%	66	2.08%	4.00%
1%	126	1.08%	8.00%	188	0.82%	18.00%	126	1.01%	1.00%
0.7%	255	0.61%	12.85%	250	0.62%	11.42%	250	0.65%	7.14%

TABLE IV: Simulation results with combined toggle patterns algorithm.

TSA	s5378			s9234			s15850		
	Toggle Pattern	SA	Delta	Toggle Pattern	SA	Delta	Toggle Pattern	SA	Delta
20%	8×7 ⊕ 16×1	20.04%	0.20%	4×4 ⊕ 8×3	19.92%	0.40%	4×1 ⊕ 8×1	19.99%	0.05%
10%	16×14 ⊕ 32×3	9.97%	0.30%	8×4 ⊕ 16×3	10.02%	0.20%	8×7 ⊕ 16×9	10.05%	0.50%
5%	32×6 ⊕ 64×1	5.01%	0.20%	16×3 ⊕ 32×2	5.01%	0.20%	16×2 ⊕ 32×3	4.98%	0.40%
2%	64×5 ⊕ 126×6	2.01%	0.50%	32×5 ⊕ 64×13	1.87%	6.50%	32×1 ⊕ 64×11	2.03%	1.50%
1%	126×3 ⊕ 188×4	1.00%	0%	64×1 ⊕ 126×3	0.95%	5.00%	64×0 ⊕ 126×1	0.96%	4.00%
0.7%	250×1 ⊕ 0	0.72%	2.85%	126×3 ⊕ 188×2	0.70%	0%	126×3 ⊕ 188×11	0.69%	1.42%

TSA	s38417			s38584			D2		
	Toggle Pattern	SA	Delta	Toggle Pattern	SA	Delta	Toggle Pattern	SA	Delta
20%	4×2 ⊕ 8×3	21.46%	7.3%	4×2 ⊕ 8×11	18.67%	6.65%	4×1 ⊕ 8×2	20.31%	1.55%
10%	8×1 ⊕ 16×3	10.03%	0.3%	8×15 ⊕ 16×1	9.94%	0.60%	16×2 ⊕ 8×5	10.24%	2.40%
5%	16×1 ⊕ 32×5	4.63%	4.70%	16×1 ⊕ 32×9	4.86%	2.80%	16×5 ⊕ 32×13	5.53%	10.60%
2%	64×6 ⊕ 126×1	2.03%	1.50%	64×2 ⊕ 126×1	2.12%	6.00%	64×1 ⊕ 126×0	2.08%	4.00%
1%	126×7 ⊕ 188×3	1.01%	1.00%	126×9 ⊕ 188×10	1.00%	0%	126×1 ⊕ 188×0	1.01%	1.00%
0.7%	188×9 ⊕ 255×11	0.7%	0%	188×2 ⊕ 250×3	0.70%	0%	188×1 ⊕ 250×1	0.70%	0%

TABLE V: Area data.

Circuit name	Area (μm^2)			Additional Area	
	Design	TG1	TG2	TG1	TG2
s5378	5.186	111	274	2.14%	5.28%
s9234	2.167	111	274	5.12%	12.64%
s15850	11.348	111	274	0.97%	2.41%
s38417	24.344	111	274	0.45%	1.12%
s38584	29.827	111	274	0.37%	0.91%
D2	21.677	111	274	0.51%	1.26%

[4] Y. Zorian, E.J. Marinissen, and S. Dey, "Testing embedded-core based system chips," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 130–143, 1998, doi:10.1109/TEST.1998.743146.

[5] E.J. Marinissen *et al.*, "Vesuvius-3D: a 3D-DfT demonstrator," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1–10, 2014, doi:10.1109/TEST.2014.7035332.

[6] J. Rabaey, *Low Power Design Essentials*. Springer Science & Business Media, 2009, doi:10.1007/978-0-387-71713-5.

[7] B. Bowhill *et al.*, "The Xeon® Processor E5-2600 v3: a 22 nm 18-Core Product Family," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 92–104, 2016, doi:10.1109/JSSC.2015.2472598.

[8] D. Czysz *et al.*, "Low-power scan operation in test compression environment," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 11, pp. 1742–1755, 2009, doi:10.1109/TCAD.2009.2030445.

[9] S. Remersaro *et al.*, "Preferred fill: A scalable method to reduce capture power for scan based designs," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1–10, 2006, doi:10.1109/TEST.2006.297694.

[10] M.A. Kochte *et al.*, "SAT-based capture-power reduction for at-speed broadcast-scan-based test compression architectures," *Proceedings of the IEEE/ACM International Symposium on Low-Power Electronics and Design (ISLPED)*, pp. 33–38, 2011, doi:10.1109/ISLPED.2011.5993600.

[11] R. Sankaralingam, R.R. Oruganti, and N.A. Touba, "Static compaction techniques to control scan vector power dissipation," *Proceedings of the IEEE VLSI Test Symposium (VTS)*, pp. 35–40, 2000, doi:10.1109/VTEST.2000.843824.

[12] K. Chakravadhanula *et al.*, "SmartScan-Hierarchical test

compression for pin-limited low power designs," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1–9, 2013, doi:10.1109/TEST.2013.6651897.
[13] D. Czysz *et al.*, "Low power compression of incompatible test cubes," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1–10, 2010, doi:10.1109/TEST.2010.5699274.
[14] S.K. Goel and E.J. Marinissen, "SOC test architecture design for efficient utilization of test bandwidth," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 8, no. 4, pp. 399–429, 2003, doi:10.1145/944027.944029.
[15] T. Waayers, R. Morren, and R. Grandi, "Definition of a robust modular SOC test architecture; resurrection of the single TAM daisy-chain," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 10–pp, 2005, doi:10.1109/TEST.2005.1584022.
[16] F. Frederick and T. McLaurin, "Design for test features of the ARM clock control macro," *Proceedings of the IEEE International Test Conference (ITC)*, pp. 1–8, 2007, doi:10.1109/TEST.2007.4437586.
[17] E.J. Marinissen and S. Deutsch, "Controlled toggle rate of non-test signals during modular scan testing of an integrated circuit," Dec. 16 2014, US Patent 8,914,689. [Online]. Available: https://www.google.com/patents/US8914689
[18] IEEE Design Automation Sub-Committee, "IEEE Std 1364-1995 Standard Hardware Description Language Based on the Verilog Hardware Description Language," 1996, doi:10.1109/IEEESTD.1996.81542.
[19] F. Brglez *et al.*, "Combinational profiles of sequential benchmark circuits," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1929–1934, 1989, doi:10.1109/ISCAS.1989.100747.