Controlling Small-Delay Test Power Consumption using Satisfiability Modulo Theory Solving

Karsten Scheibler* Matthias Sauer* Kohei Miyase† Bernd Becker*

* Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 051
79110 Freiburg, Germany
{ scheibler | sauerm | becker }@informatik.uni-freiburg.de

† Kyushu Institute of Technology
Iizuka 820-8502, Japan
k_miyase@cse.kyutech.ac.jp

***Abstract*— Exaggerated power consumption during test mode is a major issue in today's nanoscale circuits and can lead to a significant amount of overtesting and hence yield loss. We present a method that yields two-pattern tests sensitizing long paths while at the same time minimizing weighted switching activity. Experimental results on standard benchmark circuits demonstrate the applicability of the method.**

I. Introduction

Management of power consumption in test mode is mandatory for current nanoscale circuits to avoid overtesting and hence yield loss. One way to control and quantify test-pattern induced capture power consumption is the *weighted switching activity* (WSA) [1].

Previous methods based on refilling [2], [3], [4] identify unspecified X-inputs that are not needed for fault detection and specify them in order to reduce power consumption. However, as such methods depend on the initial test pattern, they are heuristic and can not achieve the total minimum. The heuristic proposed in [5] considers circuit timing and applies circuit transformation techniques to model multiple switchings of a single line. [6] controls the switching activity by using a weighted MAX-SAT formulation but is highly limited in scaling due to the explicit modelling of the weights.

We present a method based on recent advances in *Satisfiability Modulo Theory* (SMT) solvers that is able to minimize or maximize the WSA of a test pattern while at the same time guaranteeing sensitization of a given path. The method encodes the computation of the WSA-measurement directly into the SMT-instance and requires a WSA less (or more) than a given bound. At the same time, necessary conditions needed to sensitize the given path are encoded. By performing a binary search over the structural limits the test pattern pair with minimal (or maximal) WSA is obtained and extracted from the solution of the SMT solver.

Hence, in contrast to previous refilling-based approaches, the method is not limited by the quality of the given initial test pattern.

The remainder of the paper is structured as follows. Section II provides an overview over SMT solvers. The details of the proposed method are given in Section III. Experimental results are reported in Section IV. Section V concludes the paper.

II. The SMT solver iSAT

Nowadays SAT solvers are used to solve problems from many different domains. This is done by transforming the initial problem into a satisfiability problem, which means to construct a boolean formula and to ask if this formula is satisfiable or not. For example the generation of test patterns [7] is one of these applications.

Modern SAT solvers are based on the conflict-driven clause learning framework (CDCL), which is an extension of the DPLL procedure introduced in [8]. As an SMT solver *iSAT* [9] is able to process boolean combinations of linear and non-linear arithmetic constraints involving transcendental functions over real- and integer-valued variables. Especially addition and multiplication are of particular interest for our proposed method. To reason about arithmetic constraints, iSAT extends the CDCL framework by tightly integrating Interval Constraint Propagation (ICP) into it. After rewriting the input formula by a Tseitin-like transformation [10] into an equi-satisfiable conjunctive normal form (CNF), iSAT manipulates interval valuations of the real- and integer-valued variables by alternating *deduction* and *splitting* phases.

A CNF consists of a conjunction of clauses with each clause being a disjunction of atoms. Such an atom could be a boolean variable or an arithmetic constraint. Each atom is assigned to one of these values: true, false or undefined. To satisfy a formula in conjunctive normal form, every clause has to be satisfied. This is exploited in the *deduction* phase. During this phase, the solver searches for clauses in which all but one atom are false under the current interval valuation. In order to retain a chance to satisfy the formula, the remaining atom has to be true.

A conflict is found, if each atom of a clause is false. In such a case a conflict resolution procedure is called to analyze the reason of the conflict. If the conflict cannot be resolved the formula is unsatisfiable. Otherwise, a conflict clause is built from the reason of the conflict. This clause is added to the formula in order to prevent the solver running into the same conflict again (this is also called *clause learning*). Furthermore, the solver has to undo some of its decisions and deductions done so far to finally resolve the conflict.

A solution is found if at least one atom in each clause is true. The solver will stop in this case. In general, atoms containing arithmetic constraint like $x = y + z$ can only be satisfied (and evaluate to true) if the contained variables are assigned with point intervals. While this is possible with integer variables, it can not be guaranteed to reach such point intervals by ICP for continuous domains. To mitigate this problem the solver stops when all intervals have a width smaller than a certain threshold (also called *minimal splitting width*) and returns the found *approximative* solution. In such a case an a-posteriori *strong satisfaction check* [11] is done. If it succeeds the formula is known to have a solution.

Having completed the deduction phase without finding a conflict or an (approximative) solution, the solver will do a

decision by assigning true or false to a boolean variable or by *splitting* an interval of an integer- or real-valued variable. A heuristics is used to choose a variable with an interval width above the minimum splitting width. This narrowed interval may trigger new deductions and the solver continues as described above.

Lets illustrate this with a small example. Assume the following input formula with a boolean variable b and two real-valued variables x, y is given:

$$(b \oplus ((\sin(x) < 0) \wedge (y + x > 3))) \wedge \\ (b \Leftrightarrow ((y^2 < 1) \wedge (y \cdot x > 3)))$$

Because iSAT is an interval-based solver, each non-boolean variable has to be bounded. Assume in this example the initial bounds are $x, y \in [-3, 3]$. As mentioned earlier the input formula is rewritten with a Tseitin-like transformation into conjunctive normal form. This transformation may introduce new auxiliary variables, but the newly generated formula is satisfiable if and only if the given input formula is satisfiable (so called equi-satisfiablity). Because after the initial deduction phase no conflict or solution was found, iSAT does a decision, lets assume iSAT decides to assign b to false. This implies that $((\sin(x) < 0) \wedge (y + x > 3))$ has to be true. But ICP will reveal that this is impossible, because both arithmetic constraints are not satisfiable together. This results in a conflict clause (b), which implies that b has to be true. But if b has to be true, the two arithmetic constraints $(y^2 < 1)$ and $(y \cdot x > 3)$ have to be true as well. Again ICP will detect that both constraints are contradicting. So the solver was able to classify the input formula as unsatisfiable.

The following boolean and arithmetic operations are supported by iSAT:

- boolean: $\neg, \wedge, \vee, \oplus, \Leftarrow, \Rightarrow, \Leftrightarrow$
- arithmetic (unary): $\mathrm{abs}(\cdot), \sin(\cdot), \cos(\cdot), \exp(\cdot), \cdot^n, \sqrt[m]{\cdot}$ (with n, m being integer numbers: $n \geq 0$, $m > 0$)
- arithmetic (binary): $+, -, *, \min(\cdot,\cdot), \max(\cdot,\cdot)$

In the method we propose, we use only a subset of the available arithmetic operations and restrict the formula to boolean and integer variables.

Beside satisfiability checking of ordinary SMT formulas, iSAT also supports bounded model checking (BMC) [12]. BMC is often used to verify safety properties of sequential circuits. A BMC formula contains a predicate $INIT_0$ describing the initial states, a predicate $TRANS_{i,i+1}$ defining how variables change from step i to $i+1$, and lastly a predicate describing the unsafe system states $TARGET_k$. A system trace of depth k is then defined as follows:

$$\Phi_k = INIT_0 \wedge \bigwedge_{i=0}^{k-1} TRANS_{i,i+1} \wedge TARGET_k$$

BMC is used to determine whether or not a system can reach an unsafe state at a certain depth. This is done by iteratively checking $\Phi_0, \Phi_1, \ldots \Phi_k$ for satisfiability. In case of sequential circuits the three predicates are described by propositional logic. Regarding iSAT the three predicates may contain boolean combinations of rich arithmetic constraints. For this paper we concentrate on satisfiability checking of ordinary SMT formulas.

III. Overview of the method

The definition of the weighted switching activity (WSA) [1] of a test pattern pair (v_1, v_2) applied to a circuit C is based on fanout free regions (FFRs). C is considered as the disjoint union of $\mathrm{FFR}^{(1)} \cup \ldots \cup \mathrm{FFR}^{(n)}$ where n denotes the number of FFRs in C. The WSA is then computed as

$$\mathrm{WSA}(C) = \sum_{i=1}^{n} W_i \cdot S_i$$

where W_i denotes the number of gates in $\mathrm{FFR}^{(i)}$ and S_i is defined as one, iff the root gate of $\mathrm{FFR}^{(i)}$ is switching upon application of (v_1, v_2). The FFRs are directly encoded using the boolean operations supported by iSAT. Each FFR is encoded twice: FFR_1 and FFR_2 – one for each test pattern. The SMT formula is then constructed as follows:

$$\left(\bigwedge_{i=1}^{n} \left(S_i \Leftrightarrow \left(\mathrm{FFR}_1^{(i)} \oplus \mathrm{FFR}_2^{(i)}\right)\right)\right) \wedge \left(\bigwedge_{j=1}^{m} p^{(j)}\right) \\ \wedge \left(\left(\sum_{i=1}^{n} W_i \cdot S_i\right) \sim B\right)$$

with $p^{(j)}$ being the requirements to guarantee the sensitization of the target path (in our experiments we selected the 20 longest testable paths), $\sim \in \{\leq, \geq\}$ the relational operator for minimization or maximization of the WSA, and B the bound to be tested within the binary search procedure.

IV. Experimental results

We applied the method to ISCAS 85, ISCAS 89 and ITC 99 benchmark circuits. All measurements were performed on an Intel Xeon computer using one 3.3 GHz core and up to 4 GB RAM.

As the SMT-solving back-end we used iSAT. We applied the method to generate test pattern pairs sensitizing long testable paths that minimize or maximize the WSA. For the experiments, we extracted the 20 longest testable paths in each circuit using the SAT-based path generator *Phaeton* [13], [14]. The results are given in Table I. The first two columns list the name of the benchmark circuit and the total runtime in seconds needed for the application of the method. Columns "Min." and "Max." give the average minimal and maximal WSA, respectively. The numbers are defined as the percentage of the resulting WSA compared to the structural maximal WSA. The last columns show the characteristics of the generated instances. "Calls" give the total number of SMT solver calls. The average size of each generated instance in the number of Variables and Clauses is given in the last two columns.

As can be seen, our method is able to control the WSA of the circuit and yields significant differences between the minimal and maximial switching activity. At the same time, the runtimes are still reasonable given the complexity of the problem involving many optimization terms.

V. Conclusions

We presented a method which is able to minimize or maximize the weighted switching activity of a test-pattern pair sensitizing long paths. Based on an SMT-formulation, the method is able to find the true minimum or maximum WSA without being heuristic. Experimental results on standard benchmark circuits demonstrated the applicability and effectivity of the method.

Acknowledgements

Parts of this work were supported by the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) under grants BE 1176-15/2, GRK 1103 and SFB/TR 14 AVACS.

Table I
PATH SENSITIZATION USING WSA MINIMIZATION AND MAXIMIZATION

		WSA		Instance characteristics		
Circuit	Time	Min.	Max.	Calls	Variables	Clauses
c0017	0.14	31.92%	100.00%	171	136.26	195.19
c0095	0.81	24.36%	76.41%	223	642.93	870.31
c0432	24.42	10.62%	85.57%	304	3538.69	3812.70
c0499	493.65	12.20%	81.13%	313	7020.41	8727.19
c0880	11044.50	10.03%	90.41%	355	23926.20	14563.90
cs00027	0.27	71.43%	100.00%	220	204.66	299.20
cs00208	1.90	43.44%	91.60%	316	1067.84	1522.08
cs00298	2.42	14.68%	91.60%	288	1553.82	1895.67
cs00344	4.73	13.38%	91.90%	295	2169.18	2551.91
cs00349	4.76	13.32%	91.47%	295	2218.43	2589.20
cs00382	6.30	15.31%	78.47%	314	2588.38	2770.35
cs00386	2.62	13.59%	63.65%	301	1258.67	1800.56
cs00400	7.17	16.06%	76.83%	306	2737.99	2959.05
cs00420	5.64	40.46%	89.96%	358	2563.93	3617.92
cs00444	11.78	16.12%	81.03%	317	3438.87	3417.57
cs00510	8.62	30.02%	73.65%	340	3484.57	3784.34
cs00526	8.12	9.35%	92.65%	308	2671.90	3070.91
cs00641	12.08	27.39%	89.62%	372	3547.92	3785.12
cs00713	65.99	27.20%	89.98%	400	4930.90	4879.85
cs00820	6.54	9.60%	77.51%	320	2286.74	3271.01
cs00832	6.47	10.40%	75.64%	325	2278.05	3322.74
cs00838	20.85	40.00%	89.17%	400	6366.28	9506.38
cs00953	74.67	17.99%	56.10%	357	10091.70	8960.02
cs01196	95.60	16.86%	72.53%	366	10249.90	9111.72
cs01238	120.53	17.40%	68.79%	355	10538.50	9902.32
cs01488	25.26	12.36%	69.86%	380	4922.71	6622.61
cs01494	27.72	10.71%	71.11%	379	5180.80	6723.14
b01c	1.30	22.78%	88.70%	244	656.38	896.21
b02c	0.35	23.06%	85.16%	217	288.12	425.64
b03c	4.67	12.49%	96.07%	301	2668.28	2891.74
b04c	488.52	10.81%	89.57%	380	12935.60	11786.40
b05c	5504.78	15.07%	76.74%	365	22665.20	15615.00
b06c	1.18	21.25%	78.28%	239	896.21	1117.82
b07c	200.60	18.32%	91.08%	356	10992.80	9063.56
b08c	6.10	22.71%	91.72%	306	2560.71	3043.18
b09c	9.21	28.70%	94.15%	320	3139.51	3191.53
b10c	8.62	14.04%	86.29%	312	3300.65	3740.38
b11c	316.99	17.82%	81.00%	360	12838.20	10085.70
b13c	287.79	6.68%	89.00%	330	8861.80	6684.87

REFERENCES

[1] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, DAC '95, (New York, NY, USA), pp. 242–247, ACM, 1995.

[2] R. Sankaralingam, R. Oruganti, and N. Touba, "Static compaction techniques to control scan vector power dissipation," in *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, pp. 35 –40, 2000.

[3] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, L.-T. Wang, K. Saluja, and K. Kinoshita, "Low-capture-power test generation for scan-based at-speed testing," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, nov. 2005.

[4] K. Miyase, K. Noda, H. Ito, K. Hatayama, T. Aikyo, Y. Yamato, H. Furukawa, X. Wen, and S. Kajihara, "Effective IR-drop reduction in at-speed scan testing using distribution-controlling X-identification," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, 2008.

[5] S. Manich and J. Figueras, "Maximizing the weighted switching activity in combinational cmos circuits under the variable delay model," in *Proceedings of the 1997 European conference on Design and Test*, EDTC '97, (Washington, DC, USA), pp. 597–, IEEE Computer Society, 1997.

[6] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in cmos combinational circuits using boolean function manipulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 11, no. 3, 1992.

[7] A. Czutro, B. Becker, and I. Polian, "Performance Evaluation of SAT-Based ATPG on Multi-Core Architectures," in *IEEE East-West Design & Test Symposium*, September 2009.

[8] M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem Proving," *Communications of the ACM*, vol. 5, pp. 394–397, 1962.

[9] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert, "Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure," *Journal on Satisfiability, Boolean Modeling, and Computation*, vol. 1, no. 3-4, pp. 209–236, 2007.

[10] G. S. Tseitin, "On the Complexity of Derivations in Propositional Calculus," in *Studies in Constructive Mathematics and Mathematical Logics* (A. Slisenko, ed.), 1968.

[11] A. Eggers, E. Kruglov, S. Kupferschmid, K. Scheibler, T. Teige, and C. Weidenbach, "Superposition modulo non-linear arithmetic," in *Int'l Symp. on Frontiers of Combining Systems*, pp. 119–134, Springer, October 2011.

[12] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001.

[13] M. Sauer, A. Czutro, T. Schubert, S. Hillebrecht, I. Polian, and B. Becker, "SAT-based analysis of sensitisable paths," in *IEEE Design and Diagnostics of Electronic Circuits and Systems*, pp. 93–98, April 2011.

[14] M. Sauer, J. Jiang, A. Czutro, I. Polian, and B. Becker, "Efficient SAT-based search for longest sensitisable paths," in *Asian Test Symp.*, November 2011.