HARDWARE VERIFICATION BY UNIVERSAL TEST SET SIMULATIONS, SAT AND BDDS

Katarzyna Radecka
Concordia University
1455 de Maisonneuve W.
Montreal
QC H3G 1M8
Canada
kasiar@ece.concordia.edu

Zeljko Zilic
McGill University
3480 University
Montreal
QC H3A 2A7
Canada
zeljko@macs.ece.mcgill.ca

Abstract. *In this paper we consider verification of combinational circuits by test vector simulations. The simulation-based verification under the presence of a fault model uses test pattern generation approach. We consider an implicit fault model that can possibly overcome incompleteness of explicit fault models considered so far. We show that the test vector generation can be enhanced by techniques used in formal verifications: SAT- and BDD-based solutions can be combined with the vector simulations. Our method can pass useful information between these disparate approaches. Tradeoffs between the three schemes are explored.*

1 Introduction

Hardware implementation verification techniques try to assert the correctness of the circuit implementation. Verification methods based on simulations are easiest to adopt by engineers - they are shown to complement the formal verification [2], [7], usually relying on the use of BDDs, satisfiability (SAT) and automatic test pattern generation (ATPG). Underlying the simulation-based verification is a selection of a non-exhaustive *test vector set*. To verify a circuit by subjecting it to a test set, a representative *fault model* should be present. The goal is then to assert the absence of errors belonging to the fault model. In contrast, formal methods such as equivalence checking assert the equality of functions, which is equivalent to passing exhaustive tests.

In this paper we propose a complete simulation-based scheme for verifying the combinational netlist under an error fault model. Our scheme features novel use of Arithmetic Transform (AT), that is implicit in word-level decision diagrams [13], used in verification of arithmetic circuits. The transform properties guarantee the compact test pattern set for many design errors [3], consisting of various structured replacements of gates and wires in the netlist. A majority of these error classes is shown to have small AT presentation [8], and the algorithm for vector generation in that case has been given. In this work we apply these concepts towards verifying the explicit classes of gate and wire replacement errors. The same approach is extendable to faults model in [3] and elsewhere.

Our method uses in its initial stages AT-derived test vectors that easily detect design errors of small AT forms. To deal with failures that are redundant or not detected by AT vectors, we employ methods and data structures already used in formal verifications and ATPG. Towards that goal, the critical issue of the identification of redundant gate replacement errors is solved. We present an exact SAT solution to the problem, together with the filter preprocessing scheme based on the subset of the test derived by AT (shaded area in Figure 1). The second solution uses the approximate *don't care* (DC) information in the network. Based on the results of DC approximations, various SAT and ATPG simplifications are constructed in the steps to follow. We show that BDDs can be approximated, and that passing information between the processing stages can speed up SAT.

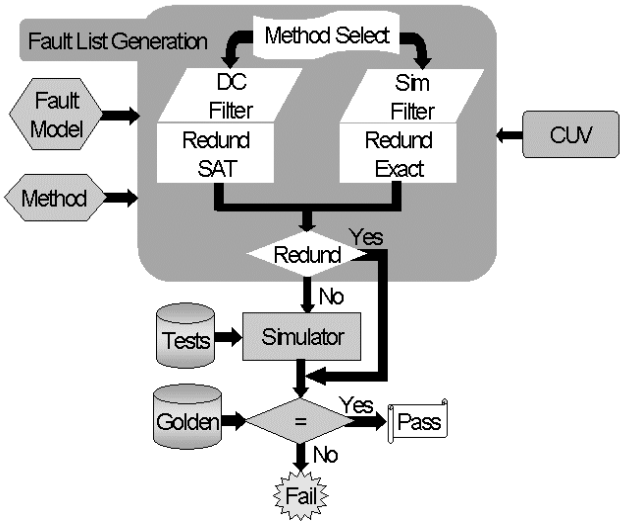

Figure 1: Verification by Simulations and SAT/BDD

1.1 Related Work

Since the underlying fault model differs significantly from the single stuck-at-value model, the usual testing approaches are not adequate. In [1] authors considered simulation-based verification that relies on a fault model including the gate and wire replacements. Authors showed that a subset of these errors could be modeled by several single stuck-at-value (s-a-v) faults, and convert each of the select replacement faults into several stuck-at faults, to which they apply ATPG and redundancy identification. Related to the overall scheme is simulation-based mutation testing [13], which is based on similar assumptions regarding the error size, but does not eliminate redundant errors. The methods that inject the faults have been recently used in verifying complex HW-SW systems, such as FPGA mapping on modern FPGA architectures [11].

The alternative approach, i.e., the formal verification, often suffers from the space explosion in data structures such as BDDs. Combining formal and simulation-based methods has been hampered by the divergence in data representations and algorithms used. The concept of filters [7] was applied towards using disparate combinational verification models, but no passing of information among the methods was facilitated. The use of s-a-v ATPG and BDDs was considered in [5] to augment test vector set for verifying safety properties. Another combination of these approaches, reported in [2], uses the identification of internal equivalent points. Then, instead of verifying the whole circuit, these internal points are checked. The method suffers from the *false negative* problem, and the BDDs are used to deal with this issue

2 Vector Generation BY AT

Arithmetic Transform is the underlying representation explored in many of the word-level decision diagrams used in formal verification by equivalence checking, as in [13]. Unlike the previous work, we use the properties of AT in simulation-based verification. The transform is defined over pseudo-Boolean functions, $f : B^n \rightarrow W$, with n binary inputs and one word-level output. These pseudo-Boolean functions actually represent the outputs of multi-output Boolean functions by word-level (W) quantities (such as unsigned integers). For example, an n-bit unsigned adder is treated as a pseudo-Boolean function with integer output values in range [0, $2^{(n+1)}$-1]. The transform can be obtained by multiplying the word-level output values with the transform matrix:

$$T_n = \begin{bmatrix} T_{n-1} & 0 \\ -T_{n-1} & T_{n-1} \end{bmatrix}, \qquad T_0 = 1, \tag{1}$$

as a matrix-vector multiplication $AT(f) = T_n * f$. The resulting transform is an integer-input polynomial with word-level coefficients:

$$AT(f(x_0, x_1 \cdots x_{n-1})) = \sum_{i_0=0}^{1} \cdots \sum_{i_{n-1}=0}^{1} C_{i_0 i_1 \ldots i_{n-1}} x_0^{i_0} \cdots x_{n-1}^{i_{n-1}}. \tag{2}$$

The faulty $\tilde{f}$ is represented as a sum of the correct output f and an error e, $\tilde{f} = f + e$. Arithmetic Transform is linear, and satisfies the equation:

$$AT(\tilde{f}) = AT(f + e) = AT(f) + AT(e). \tag{3}$$

The polynomial coefficients in the transform are referred to as the *spectrum*. Spectrum has been used to discover a number of Boolean function properties in design, test and verification [14]. The *size* of the error e is the number of spectral coefficients in *AT(e)*.

Verification by error modeling will use the test vectors to detect errors whose Arithmetic Transform is of small size in terms of number of spectral coefficients. The test generation scheme relies on the representation of pseudo-Boolean function $\tilde{f}$ by a Boolean lattice B_n. Boolean lattice structure orders all 2^n input vectors by a partial relation whereby a vector $V > V'$ if its nonzero entries are a superset of those in V' (e.g., 1011 > 1010). Consider a 2-bit adder. Spectral coefficients C_V are associated with each input vector V, as shown in Figure 2. The four nonzero coefficients are highlighted, and the corresponding points are referred to as a_1, a_0, b_1 and b_0, respectively. The ordering relation naturally forms n+1 layers in B_n, as shown in Figure 2.

The following theorem is proven in [8] in a manner similar for decoding Reed-Muller error correcting codes, or test set generation by Reed-Muller transform [4]. The theorem provides a test set among the vectors in a number of lattice layers that depends on the size t of an error spectrum.

Theorem 1: All errors that result in up to t spectral coefficients can be identified by testing $V = \sum_{i=0}^{\lceil \log_2(t+1) \rceil - 1} \binom{n}{i}$ vectors in $\left\lceil \log_2(t+1) \right\rceil - 1$ consecutive top layers of the Boolean lattice. ■

Theorem 1 provides an upper bound on the number of vectors simulated to uniquely identify this class of errors. In actual circuits, larger faults (with more spectral coefficients) will be detected. We observed [8] that high fault detection of gate replacement design errors and s-a-v faults is obtained with the constant number (up to 4) top lattice layers.

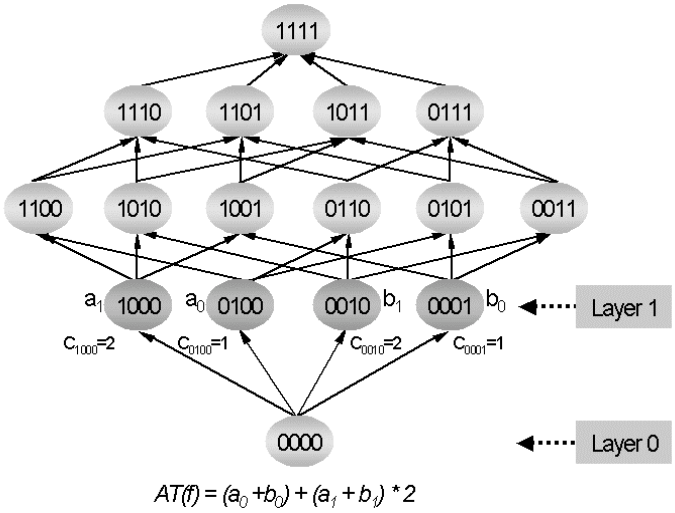

Figure 2: Lattice B_4 – 2-bit Adder Representation

The major property of AT test vectors, i.e., the detection of *all faults* of a small spectral size, allows us to quickly obtain high coverage. Unlike tackling the problem from the beginning with a deterministic scheme [1], it is beneficial to apply first AT vector simulations. Deterministic approaches can be restrained only to the last phase of verification to handle the errors that have not been detected.

3 Fault List Generation

To apply our vector generation to concrete design faults, we consider as candidate faults all possible replacements of gates in the netlist by input-compatible gates from a synthesis library. The replacements that preserve the required functionality are *redundant replacements*. Since they seriously impact the verification time, they need to be removed from the fault list. Redundancy identification is an NP-complete problem; exact methods might often be impractical. Two schemes for redundant fault identification are considered. The first roughly requires more time; the second usually demands more space. The second solution uses approximate BDD constructions by don't care subsets for preprocessing and for generating a range of approximate SAT formulations.

3.1 Redundancy Identification using SAT

Satisfiability formulation of testing problems has been elaborated in [6]. For each fault in a circuit, a conjunctive normal form (CNF), i.e., product-of-sums is constructed. If there is no solution, then the fault is redundant. The SAT formulation consists of several types of clauses. *Good circuit clauses* describe the correct operation of all circuit nodes. *Faulty circuit clauses* describe the effects of a fault on the downstream network nodes. *Active clauses* describe the activation of a fault. Finally, the *fault site* and *goal clauses* describe conditions for excitation and observation of the fault at primary outputs.

3.1.1 Exact Redundancy Identification

SAT formulation for faults, such as the replacement of a gate g with gate h, can be derived as follows. To describe requirements for activating the fault, we consider the distinguishing gate inputs, which produce $g \neq h$. The fault will be activated only for those inputs, and its polarity will not be known. We create an auxiliary node $l = g \oplus h$, and the fault location

clause will be asserted as: $l = 1$. While the fault site clauses are modified in this way, all others remain the same as in the stuck-at ATPG SAT formulation.

3.1.2 Preprocessing

Since invoking SAT might be costly, preprocessing steps reducing the number of SAT instances, are usually applied. Often, these are random simulations. In our case, vectors belong to top lattice layers. A smaller number of these layers can be used to detect all faults of size given by Theorem 1. Only if they fail, an exact SAT formulation will be invoked. Additionally, unlike random simulations, our test vector generation scheme provides the compact description of the failing vector set that can be passed to the SAT. As these stimuli don't detect any faults, they can be used to speed up the SAT process, as unsatisfying assignments.

3.2 Redundant Fault Approximations

We now outline our deterministic scheme that not only eliminates many SAT calls, but also provides information for generating easier instances of the SAT problem. We initially preprocess the replacement faults by using *don't cares* (DCs), and then apply a constrained stuck-at fault redundancy identification for faults which are more likely to be redundant. For all possible replacement faults, the *DC*-sets have to be obtained only once. They can be available for free as a by-product of the synthesis process, and are represented by BDDs.

Redundancies are caused by the DC conditions at nodes affected by faults. These are either *observability* (ODCs) or *controllability* (CDC) conditions inhibiting the error detection. By dealing explicitly with *DC*-sets, or their complements – *care sets*, the following observation holds. Each replacement h that coincides with the original gate g on a local care set, $Care_{local}$, at a given node, creates a redundant fault. A detectable gate replacement error can become redundant or hard to detect when a *DC*-set overlaps the difference between the two sets.

3.2.1 Calculating DC Approximations

Calculating *DC*-sets in a network requires determining CDCs and ODCs. Of all DC conditions ODCs are the most time- and space-consuming. The ODC sets are commonly represented by BDDs associated with each node. A good approximation is presented by Compatible Observability Don't Cares (CODCs) [12] that are obtained by a simple backwards traversal of the network. At multiple fan-out nodes, CODC of the fan-out nodes are intersected. Note that, by using DC subsets, it is guaranteed that no irredundant fault will be declared as redundant one.

3.2.2 Using S-A-V Redundancy Identification

Since we use subsets of ODCs, not all redundant replacements will be detected by this approach alone. We will reuse the DC information obtained in the first step (Section 33.2) to efficiently detect more redundancies. Our approach to identifying gate replacement errors extends the application of single stuck-at-value ATPG to redundant fault identification. Among all the possible approaches, we use the satisfiability formulation of the problem for its flexibility to deal with multiple fault models.

The good candidates for further redundancies are the gate replacements that are close to the original, measured in Hamming distance, i.e., the number of minterms in which the functions, intersected by caresets, differ:

$$d((g^{ON} \cap Care_{local}),(h^{ON} \cap Care_{local})) \leq \varepsilon . \quad (4)$$

The integer ε can be varied depending on the accuracy desired. By considering only distance one, i.e., $\varepsilon = 1$, we can guarantee that any such fault can be modeled by a single s-a-v fault. Additionally, if the replacements are within distance 1, given in cubes, the stuck-at fault redundancy identification can still be used if the replacement is monotonous, i.e., all differing outputs are of one polarity. When $g^{ON} \cap Care_{local} \geq h^{ON} \cap Care_{local}$, a fault can be detected by extending the s-a-0 ATPG. For a s-a-1 fault, the sign "$\leq$" is used instead.

In general, if the original and replaced function differ in one cube, say in k literals, then k 1-clauses will be added. Addition of each 1-clause amounts to assigning a value to a variable throughout the CNF, i.e., reducing the search space by factor of 2, up to a total reduction by a factor of 2^k. Hence, the solution to this problem is significantly quicker to find than for a single s-a-0 fault.

4 Experimental Results

Experiments on MCNC benchmarks and arithmetic circuits (adders, multipliers, ALUs and dividers) synthesized into generic Synopsys gate library GTECH show high fault coverage (> 96.7%) with test vectors obtained by AT decoding, using top 4 layers ($O(n^4)$ size). The complete (100%) redundant fault identification was obtained using our method. We built fault simulation and our redundant fault identification on UC Berkeley SIS program. Experiments were run on an Apple PowerMac G2 with two 1.25GHz PowerPC processors and 512MB of main memory, under MAC OS X v10.2 operating system.

Circuit	Size	Gate Replacements				Wire Replacements			
		Cov.	Faults	Sims	Vec.	Cov.	Faults	Sims	Vec.
ALU	24	99.3	191	516505	80	100	1570	912585	205
	32	99.5	348	1015996	84	100	2643	1618214	244
	48	99.7	736	1424568	87	100	3231	2103455	251
CLA Divid.	11x6	100	153	8720	45	100	1012	1274	68
	17x8	100	247	18834	51	95.9	2232	1683	76
	33x16	100	579	37792	75	94.1	4121	10514	97
Array Divid.	11x6	100	96	1654	17	100	825	753	7
	17x8	100	149	2147	18	100	1043	958	7
	33x16	100	293	4738	20	100	1964	1168	8

Table 1: Fault Coverage by DC Approximations

Table 1 shows the gate and wire replacement coverage for arithmetic circuits. The column in each case refers to fault coverage, the number of irredundant faults simulated, the total number of simulation runs and a number of vectors that suffice for the given coverage. The last set of numbers is obtained by counting the number of distinct vectors

that for first time detect a fault. Note that this number could be reduced through standard vector compaction methods.

In Table 2 we show that high fault coverage of the MCNC benchmarks is obtained by using the small error spectrum assumption in UTS test pattern generation. The table also reports the time and space needed to construct BDDs (needed in one of our redundancy identifications). This data is useful for comparing the cost of our method with equivalence checking by BDDs. Columns "Redund. Id" indicate times required for and coverage with our exact SAT-based redundancy identification. Finally, the last three columns report number of all faults simulated, total number of simulation runs for coverage by exact identification, as well as the total number of vectors that are sufficient for the given coverage. Total time spent in these simulations depends in general on the circuit simulation speed, but we notice that this task is amenable to massive parallelization, as both the circuits and the vectors are known in advance. We conclude that while BDDs perform worst with in space complexity, the time complexity of SAT and simulations is handled easier by preprocessing.

Circuit	DC BDD		Redund. Id			Vectors		
	size	Time	Red [s]	Total [s]	Cov. [%]	Faults	Sims	Vec.
i1	180	0.05	0	0	94.4	18	3093	10
alu2	2187	0.40	0.14	3.27	96.2	26	812	15
alu4	2148	0.94	4.51	35.7	95.9	49	6075	30
9symm	1252	0.95	0.04	0.16	97.5	6	190	2
cordic	401	0.21	0.02	0.04	92.8	28	37417	19
C499	64547	5.80	0	0.59	100	162	92466	84
C432	173829	6.07	0.15	0.31	100	167	11750	40
C1355	176390	153.7	1.99	2.26	100	68	13428	22
C1908	443558	218.4	1.71	2.11	91.2	98	650879	29
C2670	4401323	217.8	1.76	4.16	98.5	330	10088	36
C6288	∞	∞	35	55.9	100	705	66003	271
C880	30501	5.31	0.1	0.47	97.6	188	6005874	106

Table 2: Comparison of Exact, Approximate and Pre-processed Identifications

5 Conclusions and Future Work

The design error faults, including gate replacements, are significantly different in nature than stuck-at-faults. The stuck-at-fault ATPG methods cannot be directly applied. Simulations with a set of vectors derived from AT spectral properties are complemented with BDD and SAT approaches in order to refine the tests. The results from BDD-based approximations and partial simulations can be successfully passed to speed up the SAT procedures that are used as a last resort. Together, these methods present a credible simulation-based verification that can easily be automated, parameterized by the error size,

and performed by means of orthogonal, but cooperative verification methods. This approach can be extended to other error models.

As the proposed method uses data representation and algorithms widely used in formal methods, the simulation approach can be readily applied in combination with equivalence checking, for which we constructed a generalization of AT representation [8]. Further, as the same representations possess properties useful in verifying datapaths within allowed imprecision [9]. Further work on the combination of the formal and simulation-based methods could extend the reach of the methods considered here.

References

[1] H. Al-Assad and J. P. Hayes, "Design Verification via Simulation and Automatic Test Pattern Generation*", Intl. Conf. on Computer Aided Design*, 1995, pp. 174-180.

[2] J. Burch and V. Singhal, "Tight Integration of Combinational Verification Methods", *Intl. Conf. on Computer Aided Design,* 1998, pp. 570-576.

[3] D. van Campenhout, H. Al-Assad, J.P. Hayes, T. Mudge, R. Brown, "High-Level Design Verification of Microprocessors via Error Modeling", *ACM Trans. Design Automation of Electronic Systems*, 1998, 3(4), pp.581-599.

[4] T. Damarla, and M. Karpovsky, "Fault Detection in Combinational Networks by Reed-Muller Transform", *IEEE Transactions on Computers*, 38(6), pp. 788-797, Jun. 1989.

[5] M.K. Ganai, A. Aziz, and A. Kuehlman, "Enhancing Simulation via BDDs and ATPG", *Proceedings of DAC,* 1999, pp. 297-301.

[6] T. Larrabee, "Test Pattern Generation using Boolean Satisfiability", *IEEE Trans. CAD of Integrat Circuits and Systems*, 1992, 11(1), pp. 4-15.

[7] R. Mukherjee, J. Jain, K. Takayama, M. Fujita, J.A. Abraham, and D.S. Fussell, "An Efficient Filter-Based Approach for Combinatorial Verification", *IEEE Trans. on CAD of Integrated Circuits and Systems*, 18(11), 1999, pp. 1542-1557.

[8] K. Radecka and Z. Zilic, "Using Arithmetic Transform for Verification of Datapath Circuits via Error Modeling", *Proceedings of VLSI Test Symposium*, *VTS 2000*, pp. 271-277.

[9] K. Radecka and Z. Zilic, "Arithmetic Transforms for Verifying Compositions of Sequential Datapaths", *Proceedings of International Conference on Computer Design*, *ICCD 2001*, pp. 348-353.

[10] K. Radecka and Z. Zilic, "Specifying and Verifying Imprecise Arithmetic Circuits by Arithmetic Transforms", *International Conference on Computer-Aided Design, Digest of Technical Papers*, ICCAD 2002, pp. 83-86.

[11] B. Ratchev, M. Hutton and B. van Antwerpen, "Logic Synthesis and Mapping: Verifying the Correctness of FPGA Logic Synthesis Algorithms", *Proceedings of ACM International Symposium on FPGAs*, 2003, pp. 127-135.

[12] H. Savoj and R. Brayton, "The Use of Observability and External Don't Cares for the Simplification of Multilevel Logic Networks", In *Proceedings of International Conference on Computer Aided Design,* 1990, pp. 297-301.

[13] C. Scholl, B. Becker, and T.M. Weis, " Word-level Decision Diagrams, WLCDs and Division", In *Proceedings of International Conference on Computer Aided Design,* 1998, pp. 672-677.

[14] M. A. Thornton, R. Drechsler and M. D. Miller, "*Spectral Techniques in VLSI CAD*", Kluwer Academic Press, 2002.

[15] P. Vado, Y. Savaria, Y Zoccarato, and C. Robach, "A Methodology for Validating Digital Circuits with Mutation Testing", *Proceedings of ISCAS,* 2000, pp. I-343-346.