Pseudo-exhaustive Testing of Sequential Circuits

Bassam Shaer
University of Minnesota
Department of Electrical & Computer Engineering
bshaer@d.umn.edu

Sami A. Al-Arian
University of South Florida
Computer Science & Engineering Department
alarian@usf.edu

David Landis
The Pennsylvania State University
Center for Design and Computing
dllcdc@engr.psu.edu

Abstract

A new sequential circuit partitioning algorithm is introduced which enhances pseudo-exhaustive testing. Our PIFAN algorithm is based on an analysis of Primary Input cones and FANout values. Results are presented which show that PIFAN offers significant reductions in hardware overhead and test time when compared to alternative partitioning algorithms.

1. Introduction

Because exhaustive testing is impractical for large VLSI circuits, partitioning which allows exhaustive test of sub-circuits offers an attractive alternative. Goel [1] observed that with each doubling of the number of gates in a circuit, the cost of testing increases as the square of the previous cost.

McCluskey [2] proposed pseudo-exhaustive testing as an alternative, where the circuit is partitioned into p sub-circuits. Each partition has an upper bound s on the number of inputs, while the total number of edges between the partitions is minimized. In general, a wide variety of automated techniques for circuit partitioning and testability enhancement are available, and their impacts on the Circuit Under Test (CUT) differ greatly. However, all efficient methods for general partitioning and test vector generation for digital circuits require significant amounts of Design for Testability (DFT) hardware [3]. For highly regular circuits such as ROMs, RAMs, and PLAs, efficient Built-In Self-Test (BIST) methods have been developed based on their distinct fault assumptions [4, 5].

In the context of this paper, partitioning is the process of creating a logic hierarchy for efficiently carrying out tests. Partitioning techniques can be broadly separated into two categories: the *invasive approach* and the *noninvasive approach* [6]. The *invasive approach* isolates one portion of a circuit from another by inserting DFT elements [2,3,7,8,9,10]. All the invasive approaches pay area overhead penalties associated with the DFT circuitry and suffer performance degradation due to the extra time delay caused by signals propagating through the added hardware. The *noninvasive approach* achieves subcircuit isolation without requiring extra hardware [2]. In this case there is low area overhead and no performance degradation because extra (test) hardware is not inserted in the signal paths.

Some techniques utilize both invasive and noninvasive approaches [11]. The invasive portion of this combined approach uses DFT circuitry to increase testability and achieve global partitioning among functional units, while the noninvasive partitioning can be regarded as local partitioning. Sensitized partitioning is used to partition each functional unit for pseudo-exhaustive testing to minimize hardware overhead and performance degradation.

In this paper, we develop a partitioning algorithm (PIFAN) which is based on an analysis of Primary Input cones and FANout values. Before applying our algorithm to sequential circuits, they must be pre-processed to transform them into combinational circuits. This is done by removing every D flip flop, replacing its D-input by a primary output, and replacing its Q-output by a primary input. Next, the circuit is levelized. We then apply the PIFAN algorithm [9] to the levelized combinational circuit until all nodes can be tested pseudo-exhaustively. Finally, all flip-flops are re-inserted and modified to form a full-scan configuration except for those (n) flip-flops that will be modified to form a test cell chain. This test cell chain will be used to scan in the test vectors to control all partitioning points (test data - TD) and to scan out responses to observe different partitioning points in the combinational portion of the circuit. The following section introduces the scanable model of the D flip-flop and test cell, and gives a detailed description of our procedure for partitioning sequential circuits.

2. Problem Approach

The approach followed in this work is to take a gate/block level VLSI sequential circuit and partition it into smaller sub-circuits. Each partition must not be dependent on too many inputs, and the amount of control

0-7695-0104-4/99 $10.00

logic between the partitions must be as small as possible. By exhaustively testing each partition, testing the added logic, and also testing all D flip-flops, we pseudo-exhaustively test the whole circuit. We map the VLSI circuits into graph-theoretic models in order to represent them in a form that can be algorithmically processed. As in [9], a circuit can be represented by a directed graph containing two types of nodes. The *Type I* nodes are the square nodes corresponding to registers (scanable D flip-flops as shown in Figure 1a) and inputs or outputs of the circuit. The *Type II* nodes (circle nodes) represent the gates/blocks of the circuit.

Based on the graph model, the partitioning problem can be formulated in terms of the following requirements. Given a directed acyclic graph, we partition the graph into sections by adding triangular nodes. A solid triangular node represents a TC1.m test cell that will act as a bypass node in the normal mode and a shift register element in the test mode. This node type will be used to control and observe different partitioning points. The hollow triangular node, on the other hand, represents a TCM test cell, which is a 2-to-1 multiplexer that will be used to control a specific partitioning point. Note that each partition should not contain circle nodes with a primary input degree that is larger than the upper bound. To minimize the overhead, the total number of triangular nodes added to the graph after it has been completely partitioned should be as small as possible. Our test cell TC1.m, shown in Figure 1b, has the same configuration as the scanable D flip-flop except that the 2-to-1 multiplexer is replaced by an m-to-1 multiplexer to observe (m-2) partitioned points within the combinational part of the circuit. These test cells will use existing D flip-flops as their latches to reduce the overhead. The normal data input (D) of a latch will be connected to the normal data input (NDI) of a test cell and the normal output of the latch (Q) will be connected to the NDO output of the test cell.

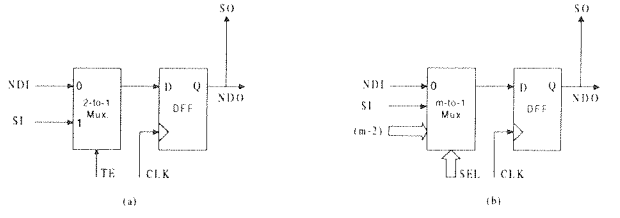

Figure 1. (a) Scanable D flip-flop, (b) TC1.m cell

We propose using a chain of n TC1.m test cells (TC1-chain), where n is the maximum number of partitioned points that any primary or secondary output or partitioned point depends on. The TC1-chain delivers the pseudo-exhaustively generated test patterns to control the TCM test cells, and scans out the output responses to be observed at the external scan out (SO). We also propose to modify the scanable D flip-flop by adding a 2-to-1 multiplexer at its output, as shown in Figure 2. One of the inputs to the multiplexer is connected to the normal Q output of the latch and the other input is connected to the test data (TD) input. All TD inputs of the modified flip-flops are connected to the TC1-chain to receive the input test patterns. The motive for doing this is to make test pattern delivery easier and to insure complete concurrent testing of all combinational blocks connected to the inputs of all D flip-flops. The modified D flip-flops are configured in a full-scan mode in order to capture the responses and scan them out for observation.

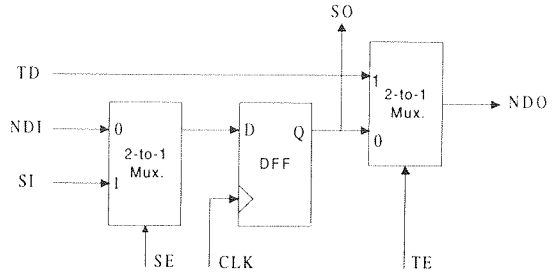

Figure 2. Modified scanable D flip-flop

We modify n arbitrary flip-flops to form the TC1-chain that is used to deliver test patterns to all partitioned points and to capture and scan out output responses. All other flip-flops will form a scan path to deliver test stimulus for the secondary inputs and capture the secondary output responses.

3. Partitioning Sequential Circuits

The PIFAN Partitioning algorithm is based on the concept of partitioning the nodes in a graph tree that have a high Primary Input cone (PI) value. The goal is to reduce the primary input cones to n primary inputs (where n is known). It first partitions all nodes with highest Fan Out (FO) values and primary input cone greater than one, then it processes all other nodes starting from first level nodes. The rationale for partitioning the nodes with a high fanout value is that these nodes increase the number of primary input cones of all the nodes that are connected to them. Thus, partitioning at high fanout nodes provides the largest reduction in input cones because the greatest number of nodes are affected. The PIFAN algorithm has four phases: circuit pre-process, circuit process, circuit post-process and testing process.

3.1. Circuit Pre-Process

The circuit pre-process consists of the following steps:

1. Transform sequential into combinational circuit.
2. Create circuit description file with all gate names in levelized order.
3. Run PIFAN-PT (Primary Inputs FANout Processing Tool).

Figure 3 shows the annotated graph model of the S27 benchmark circuit after running PIFAN-PT, and indicates the (PI, FO) values for each node in the graph model.

3.2. Circuit Process

The PIFAN partitioning algorithm starts by setting the maximum number of primary inputs that a partition may depend on to n, then it processes the circuit to:

1. Find all nodes with PI and FO values greater than 1. From this list it identifies all nodes with the maximum fanout FO, which are not located next to a primary output. These nodes are selected as partitioning points, however, the algorithm makes sure not to partition the output of an inverter or a buffer. Partitioning the input source of an inverter or buffer takes care of its output fanouts.
2. Modify the circuit description file for every partitioning point by adding a primary input and a primary output unless the node is already an external output.
3. Run the PIFAN-PT tool to generate the new fanout values and primary input cones and place results in a new file (CIRCUIT-NAME.PFL).
4. Search the CIRCUIT-NAME.PFL file for the first node with a PI $\geq$ n and FO $\geq$ 2. If a node has PI = n, then process the node; partition all nodes having a PI value of n and before any node with a PI > n. Else if node has a PI > n, then process node, partition it and go to step (2).
5. If no more nodes in the CIRCUIT-NAME.PFL file have PI > n, then search the file CIRCUIT-NAME.ALL that contains all nodes and their PI's values (which is updated with every PIFAN-PT run) for the first node with PI > n. If found, process node and partition it and go to step (2), else stop.

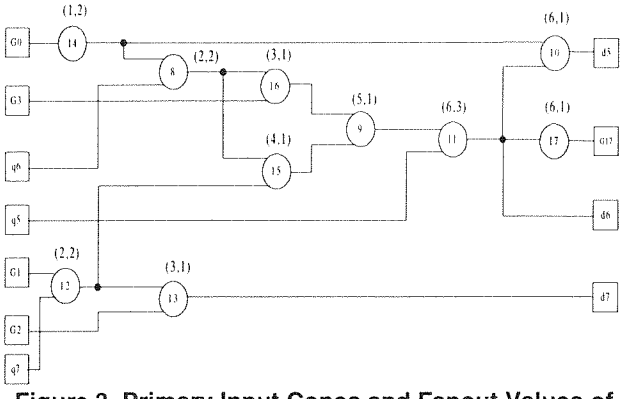

Figure 3. Primary Input Cones and Fanout Values of the S27 Benchmark Circuit

3.3. Circuit Post-Process

After the modified sequential circuit has been partitioned, the next phase is to add the test cells and configure the D latches to form a full-scan chain. We first find the maximum number of secondary inputs and partitioned points that one partition depends on. This is the minimum number of TC1.m test cells that need to be used to form the TC1-chain configuration. The TC1-chain will be used to scan-in test patterns and to scan-out responses for observation. For the other cuts, the TCM test cell is added to control the corresponding partitioning points. We minimize overhead by using some of the existing latches to form the TC1.m test cells. For all other latches, we propose the use of the modified scan cell shown in Figure 2. Each modified scan cell has a test data (TD) input that is connected to one NDO of the TC1-chain. The TC1-chain delivers the test vectors to all the modified flip-flops. To calculate the overhead added by the partitioning algorithm, we use the transistor count values given in Table 1.

Table 1. Devices and their transistor counts

Device Type	# of Transistors	Device Type	# of Transistors
Inverter	2	TCM	4
Buffer	4	4-to-1 Mux	8
NAND	2n	8-to-1 Mux	16
NOR	2n	D Latch	22
AND	2n + 2	Modified D Latch	26
OR	2n + 2	TC1.2 Cell	22
XOR	10	TC1.4 Cell	26
Static DFF	18	TC1.8 Cell	34

3.4. Testing Process

Exhaustive test vectors are generated to test each partitioned point, primary output, and secondary output in the circuit. These test vectors will be applied to the primary inputs and shifted through the TC1-chain configuration to control all secondary inputs and partitioned points. By capturing the response through the full-scan configuration we can observe the response of the circuit concurrently. Also, by changing the select lines of the TC1-chain, we can observe all partitioned points in the combinational blocks.

4. Analysis of Results

We have applied the PIFAN partitioning algorithm to the ISCAS89 sequential benchmarks circuits [12]. Table 2 shows the result of our algorithm and summarizes the data we have obtained. In all these experiments, we first set the PI value n to an acceptable limit for pseudo-exhaustive testing, then we ran the PIFAN algorithm. If the primary input dependencies of the unmodified circuit allow pseudo-exhaustive testing, then no partitioning is needed. The next phase was to construct the TC1-chain configuration and then generate the pseudo-exhaustive test patterns. For example, when applying our algorithm to the s27 and the s298 benchmark circuits, no partitioning

was needed. For the s27 circuit we used its D latches to form the TC1-chain to scan in the test patterns and to observe the secondary outputs. The circuit s27 can be tested exhaustively using $2^6 = 64$ test patterns. However, after partitioning s27 by adding three partitioning points, the circuit can be tested pseudo-exhaustively using only 16 test vectors.

We ran the PIFAN algorithm on several sequential circuits using two different values of n. In the first run we used n = 8, and in the second we used n = 12. Both results are shown in Table 2. Table 3 compares the results obtained using the PIFAN algorithm with the results given in [12]. The test patterns given in [12] were not created using an automatic test pattern generation algorithm; they were randomly selected.

Table 2. Results of Applying the PIFAN Algorithm to Sequential Circuits

Circuit Name	Max PI (n)	# of Partitioned Points	Overhead	# of Test Patterns
s27	3	3	22%	16
	8-12	0	0	64
s298	8	0	0	256
	12	0	0	256
s444	8	7	4.6%	512
	12	2	1.32%	4,608
s641	8	21	7.55%	768
	12	15	5.4%	7,168
s953	8	86	22.8%	1,792
	12	49	12.9 %	12,288
s1238	8	90	19.4 %	1,536
	12	31	6.6 %	8,704
s9234	16	106	2.4%	458,752

Table 3. Comparison between [Brgl89] and PIFAN

Circuit Name	Brgl89		PIFAN Algorithm		
	Fault Coverage	# of Test Patterns	n Value	# of C/O Points	#of Test Patterns
s27	100	31	3	3	16
s298	100	554	8	0	256
s444	97.05	835	8	7	512
s641	99.14	1,199	8	21	1,024
s953	100	2,999	8	86	1,792
s1238	94.75	3,001	8	90	1,536
s9234	89.65	108,774	16	106	458,752

5. Conclusion

The Primary Input cones and FANout partitioning algorithm was described and shown to efficiently partition large combinational and scan-based sequential circuits for pseudo-exhaustive testing. ISCAS89 benchmark circuits containing up to 5,597 gates have been successfully partitioned using PIFAN, and when compared with other partitioning algorithms, PIFAN was superior. Future work to improve the PIFAN algorithm includes automating the determination of an optimal PI value *(n)*. Additional work is also needed to minimize the added hardware overhead and to evaluate and minimize the impact on system timing. Also, further research is needed to develop a concurrent testing strategy so that as many sub-circuits as possible can be tested in parallel.

6. REFERENCES

[1] Goel, P. "Test Generation Costs Analysis and Projections," *The 17th Design Automation Conference, ACM/IEEE*, June 1980, pp. 77-84.

[2] McCluskey, E. J. and S. Bozorgui-Nesbat, "Design for Autonomous Test," *IEEE Trans. on Computer*, Vol. C-30, No 11, Nov. 1981, pp. 866-875.

[3] Udell, J. G. Jr., and E. J. McCluskey, "Efficient Circuit Segmentation for Pseudoexhaustive Test," *Proc. Int. Conf. on Computer-Aided Design*, 1987, pp. 148-151.

[4] Kinoshita, K. and K. K. Saluja, "Built-In Testing of Memory Using an On-Chip Compact Scheme," *IEEE Trans. on Computer*, Vol. C-35, Oct. 1986, pp. 862-870.

[5] Liu, C. Y. and K. K. Saluja, "BIST-PLA: A Built-In Self-Test Design of Large Programmable Logic Arrays," Proc. *Design Automation Conf.*, 1987, pp. 385-391.

[6] Su, Chau-Chin, "Computer-Aided Design of Pseudoexhaustive Test for Data Paths," Doctoral Dissertation, The University of Wisconsin - Madison, Jul. 1990.

[7] Konemann, B. et al., "Built-In Logic Block Observation Techniques," *Proc. 1979 Test Conf.* pp. 37-41, Cherry Hill, N. J., Oct. 1979.

[8] Komonytsky, D., "LSI Self-Test Using Level Sensitive Scan Design and Signature Analysis," *Proc. Int. Test Conf.*, 1982, pp. 414-424.

[9] Shaer, B., Al-Arian, S. and Landis, D., "Partitioning Algorithm to Enhance VLSI Testability," *Proceeding of 36th ACM Southeast Conference*, 1998, pp. 121-129.

[10] Shperling, I., and E. J. McCluskey, "Circuit Segmentation for Pseudo-Exhaustive Testing via Simulated Annealing," *Proc. Int. Test Conf.*, 1987, pp. 58-65.

[11] McCluskey, E. J., "Built-In Verification Test," *Proc. Int. Test Conf.*, 1982, pp. 183-190.

[12]Brglez, F., D. Bryan, and K. Kozminski, "Combinational Profile of Sequential Benchmark Circuits," *IEEE Int'l Symp. on Circuits and Systems*, 1989, pp. 1929-1934.