Partial Scan Design Based on Circuit State Information

Dong Xiang Srikanth Venkataraman W. Kent Fuchs Janak H. Patel
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

State information of a sequential circuit can be used to evaluate the complexity of test generation. The ratio of valid states to all the states of the circuit is an important indicator of test generation complexity. Using valid states obtained via logic simulation, a testability measure based on the density of encoding is proposed for scan flip flop selection. A second testability measure based on the test generation state information is also presented and used to select scan flip flops. Cycles are broken selectively on the basis of the circuit state information. Good fault coverage and test efficiency are obtained when fewer scan flip flops than the minimum cut set are selected. Experimental results are presented to demonstrate the effectiveness of the method.

1 Introduction

The complexity of sequential circuit ATPG is prohibitive for large or highly sequential circuits. A variety of partial scan design methods have been developed to reduce the complexity of the problem. They are usually classified into the following three categories: structure-based [4,5,7,8], testability-measure-based [6,7,9], and test-generation-based methods. Additional methods include layout-driven, timing-driven and retiming-driven methods. In order to reduce test application time, techniques have been developed for ordering scan flip flops or test sequences after scan elements have been selected. Recent work has also shown that state information and partial scan design can be used for fault diagnosis [13].

The structure-based approaches to scan selection utilize the fact that test generation complexity is exponential in the size and the number of the cycles. Typically, a circuit has significantly fewer valid states than all the possible states. For a cycle with size n, if 2^n states are valid states, no backtracking is needed in test generation because of the cycle. If only a few states of the cycle are valid, then many backtracks may be caused by the cycle during test generation. An effective test generator may traverse only a few invalid states that cause backtracks. Therefore, the typical approach of cutting all the cycles may not be the best approach to the partial scan design problem, since some of the cycles do not have a significant influence on test generation complexity.

Marchok et al. presented a series of experimental results [1] to evaluate the complexity of test generation for sequential circuits. Their experimental conclusion is that the density of encoding (ED for short) is the key indicator of sequential circuit ATPG complexity for a circuit,

$$ED = \frac{V}{2^n} \tag{1}$$

where V is the number of valid states, and n is the number of flip flops in the circuit. However, for most large circuits, the density of encoding is far less than 1. We utilize a testability measure based on the valid states and on separate cycles.

This paper introduces three separate schemes for choosing scan flip flops according to state information. The first scheme derives the valid states via logic simulation or from the state transition table. The logic simulation obtains almost all the valid states for some circuits. However, when the number of flip flops in the circuit or the number of valid states is very large, the method only obtains a subset of all the valid states. However, the partial valid state set still provides enough information for effective scan flip flop selection. The second scheme uses a sequential circuit test generator to create tests for a subset of the faults in the circuit. The invalid states traversed by the test generator in the ini-

Supported in part by the National Science Foundation of China, in part by ARPA under grant DABT63-95-C-0069, in part by the Semiconductor Research Corporation under contract SRC 95-DP-109, and in part by the Joint Services Electronic Program under grant N00014-96-1-0129.

33rd Design Automation Conference ®

DAC 96 - 06/96 Las Vegas, NV, USA

tial stage are recorded. These states indicate unsatisfiable signal requirements. Finally, tests of the fully scanned circuit are generated. The pseudo primary input (PPI) portions of the tests indicate signal requirements of the flip flops during test generation. These tests are used to guide scan flip flop choices.

2 Background

Definition 1 *A state is an assignment of boolean values* $\{0, 1\}$ *to the outputs of the flip flops. The reset state is a state that can be reached from any state of the circuit.*

Definition 2 *A state is a valid state if it is reachable from the reset state; a state is an invalid state if it is not reachable from the reset state. A valid state can be justified to the primary inputs, while an invalid state will cause backtracks in the process of test generation.*

Definition 3 *Assume a state is an* n*-tuple* $(v_1, v_2, \ldots, v_n)$*, where* n *is the number of flip flops in the circuit, a partial state* $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$*, where* i_1*,* i_2*, ...,* $i_k \in \{1, 2, \ldots, n\}$ *and* $k < n$*. A partial state is called a partial valid state if the corresponding state is valid; it is called a partial invalid state if the corresponding state is invalid. State mapping maps a state to all the cycles in the circuit, where each cycle contains a flip flop subset.*

Definition 4 *The vertices of the s-graph of a sequential circuit are the flip flops of the circuit. There is an edge between two vertices if there is one path between them and no other flip flop in the path.*

Definition 5 *Density of encoding [1] of a circuit is defined as* $\frac{V}{2^n}$*, where* n *is the number of flip flops in the circuit, and* V *is the number of valid states of the circuit.*

The number of valid states V for a circuit with n flip flops is usually much less than 2^n. When $\frac{V}{2^n}$ is much less than 1, the test generator may frequently justify invalid states. An invalid state causes backtracks during test generation.

We present three separate schemes for choosing scan flip flops. The first scheme selects scan flip flops based on the valid states. The second scheme selects scan flip flops based on the states the test generator traversed during the initial stage. The third scheme chooses scan flip flops based on the PPI test portion of the fully scanned circuit.

Table 1: Simulation Results for the 89ISCAS Circuits

circuits	PIs	FFs	states	vectors
s27	4	3	5	10000
s208	11	8	254	10000
s298	3	14	90	10000
s344	9	15	1284	10000
s349	9	15	805	10000
s382	3	21	33	10000
s386	7	6	17	10000
s400	3	21	30	10000
s420	19	16	1543	10000
s444	3	21	25	10000
s510	3	6	1	10000
s526	3	21	35	10000
s526n	3	21	37	10000
s641	35	19	77	10000
s713	35	19	876	10000
s820	18	5	23	10000
s832	18	5	21	10000
s838	34	32	3957	10000
s838.1	34	32	3957	10000
s953	16	29	882	10000
s1196	14	18	4157	10000
s1238	14	18	4657	10000
s1423	17	74	4905	10000
s1488	8	6	64	10000
s1494	8	6	64	10000
s5378	35	179	8207	10000
s9234	36	211	4843	5000
s13207	62	638	3073	5000
s13207.1	62	638	3161	5000
s15850	77	534	363	5000
s35932	35	1728	344	1000
s38417	28	1636	56	5000
s38584	38	1452	61	5000

3 Valid States for Flip Flop Selection

We use logic simulation to obtain as many valid states as possible. Logic simulation can be finished in linear time for each vector. If the synchronizing sequence or the reset state is given, we can start logic simulation from the reset state. Otherwise, we start logic simulation with all the flip flops unspecified. Some circuits are not so easily to initialize. We can randomly set the sequential circuit into a specific state although it may be an invalid state. If a circuit is effectively designed, after a few vectors have been applied, the state of the circuit is typically transformed to a valid state.

During logic simulation, when a new state still cannot be reached after enough patterns have been applied, a previous recorded state which occurs the least number of times is chosen to continue logic simulation. In Algorithm 1, T

is the number of vectors that have been simulated. L is the number of consecutive vectors that generate no new state. S is the valid state set. n_Q is the number of times the state Q occurs during logic simulation. $f(Q_i, t_i)$ is the number of times the $2-$tuple (Q_i, t_i) occurs, where Q_i is a state, and t_i is a vector. *limit1, limit2, limit3* are chosen to give bounds for the number of occurences for a (state,test) tuple, the number of vectors applied, and the number of acceptable consecutive vectors that generate no new state. Values of the three limits used by us are 10, 10000 and 50 respectively.

Algorithm 1 (valid state set via logic simulation)

1. Assume R is the reset state of the circuit, $Q_i \leftarrow R$, go to step 3; otherwise, if the synchronizing sequence is given, perform logic simulation for each pattern on the circuit, the final state is the reset state; go to step 3;
2. If no reset state or synchronizing sequence of the circuit is available, randomly set the circuit to a state, do
   (a) generate a random vector;
   (b) perform logic simulation using the vector on the circuit;
   (c) repeat (a), (b) until a specified number of vectors have been simulated.
3. $S \leftarrow \{Q_i\}$, where Q_i is the current state of the circuit;
4. Generate a random vector t_i,
   (a) if $f(Q_i, t_i) > limit1$, then go to step 5;
   (b) perform logic simulation on the circuit using t_i to generate the next state Q_{i+1}. $f(Q_i, t_i) \leftarrow f(Q_i, t_i) + 1$;
   (c) if Q_{i+1} is not in S, then $L \leftarrow 0$, $S \leftarrow S \cup \{Q_{i+1}\}$, $n_{Q_{i+1}} \leftarrow 1$, $Q_i \leftarrow Q_{i+1}$, go to step 4;
   (d) if the state $Q_{i+1} \in S$, $L \leftarrow L + 1$, $n_{Q_{i+1}} \leftarrow n_{Q_{i+1}} + 1$;
   (e) $T \leftarrow T + 1$;
5. If $T > limit2$, end the procedure;
   else if $L < limit3$, then $Q_i \leftarrow Q_{i+1}$, go to 4;
   if $L > limit3$, choose a state Q_i from S, where Q_i occurs the fewest times. When more than one of these states exists, choose the first state Q_i. Go to step 4.

Table 1 presents logic simulation results for ISCAS89 circuits. Some of the circuits have too few valid states. The others generate enough states. All the logic simulation results are obtained in no more than one minute using a SPARC-20.

Table 2: Valid States of S400

00011110000000000000	00011110000000000001
11011110000000000000	00011110000000010000
11011110000000000001	10011110000000000000
10011110000000010011	01011110000000000001
01011110000000000011	10011110000000000001
01011110000000000000	11011110000000010001
01011110000000010000	01011110000000010011
11011110000000010000	01011110000000100001
01011110000000100000	11011110000000010011
01011110000000010010	01011110000000010001
10011110000000000011	11011110000000000010
01011110000000000010	00011110000000010001
10011110000000010000	00011110000000000010

4 Partial Scan Flip Flop Selection via Valid States

Our analysis shows that testability of a circuit is worse if there are more flip flops with unchanging values, or if there are a couple of cycles with single or few states. The state information obtained is used to direct scan flip flop choices. Table 2 presents the 30 valid states of s400 after 10000 vectors have been simulated. Most of the flip flops have unchanged values for all the states. 7 of the 21 flip flops have changed values. That evidence suggests that we should not simply break all of the cycles, but that we should break them selectively.

According to the cycle identification algorithm, there are 18 cycles in s400. According to the valid state set, 7 cycles assume single state and 1 cycle has all the possible states. These results are used as shown below to guide scan flip flop selection.

Each state is mapped to the cycles. The testability measure for each flip flop n is evaluated as,

$$T(n) = \sum_{c_i} T(n, c_i) \tag{2}$$

where c_i represents all the cycles containing flip flop n.

$$T(n, c_i) = \begin{cases} \frac{2^{k_i}}{f(c_i)} & \text{if } k_i \leq 15 \\ k_i \cdot \frac{2^{15}}{f(c_i)} & \text{if } k_i > 15 \end{cases} \tag{3}$$

where c_i represents all the cycles through the flip flop, and $f(c_i)$ is the number of different states according to the state set obtained above. k_i is the size of the cycle c_i. 15 is an empirical constant.

The testability measure reflects the possibility of entering invalid states in the process of test generation. The testa-

bility improvement potentiality (TIP) measure for each flip flop is used to evaluate potential testability improvement of the flip flop if it is chosen to be a scan flip flop. The TIP measure for each flip flop n is given in equation 4.

$$TIP(n) = \sum_{c_i} (T(n, c_i) \cdot k_i) \quad (4)$$

The flip flop with the largest TIP measure is chosen each time as a scan flip flop. After a scan flip flop is selected, testability measures and TIP measures of the remaining flip flops are updated.

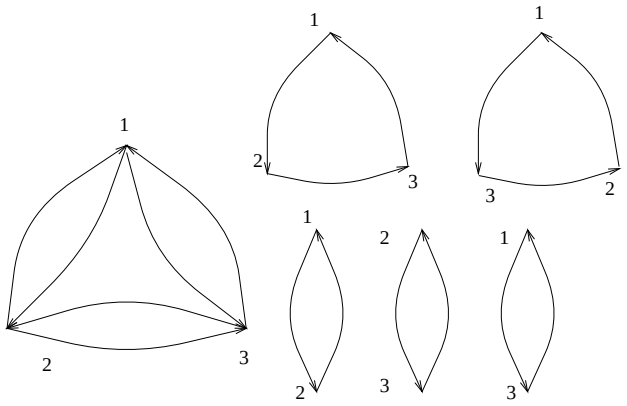
Algorithm 2 (scan flip flop selection using valid states)

1. For each cycle c_i, $f(c_i) \leftarrow 0$;
2. For each state Q_i in the state set, do
   (a) state Q_i maps a partial valid state $(v_1, v_2, \ldots, v_k)$ for each cycle c_i;
   (b) if $(v_1, v_2, \ldots, v_k)$ is a new partial valid state for cycle c_i, $f(c_i) \leftarrow f(c_i) + 1$;
3. For each flip flop, calculate the testability measure and the TIP measure using equations $2 - 4$.
4. Choose the flip flop n with the largest TIP as a scan flip flop. For each cycle c_i, and each other flip flop n_1 in the cycle c_i, the testability measure for n_1 is decreased by $T(n, c_i)$ and the TIP measure is decreased by $k_i \cdot T(n, c_i)$.
5. If no more scan flip flops are needed, end the algorithm. Otherwise, go to step 4.

Example 1: Assume the s-graph of a finite state machine is shown in figure 1(a). The valid state set is $\{100, 101, 110, 111\}$. Figure 1(a) is the s-graph and figure 1(b) shows its cycles. We want to choose a scan flip flop from the 3 flip flops. According to the 4-state valid state set, the states for cycles $\{1, 2, 3\}$, $\{1, 3, 2\}$, $\{1, 2\}$, $\{2, 3\}$ and $\{1, 3\}$ are 4, 4, 2, 4 and 2 respectively. Testability measures for flip flops 1, 2 and 3 are 8, 7 and 7. TIP measures for flip flops 1, 2 and 3 are 20, 18 and 18 respectively. Therefore, flip flop 1 is selected as the scan flip flop.

5 Scan Flip Flop Selection Based on Test Generation State Information

We shall introduce techniques for scan flip flop selection using state information obtained by test generation. We utilize two schemes to generate circuit state information. The first scheme utilizes state information of a test generator in the initial stage. The second scheme generates tests for the fully scanned circuit. Algorithm 3 chooses scan flip flops according to the states traversed by the deterministic test generator.



(a) S-graph (b) cycles

Figure 1: S-graph and scan flip flop selection

In Algorithm 3, S is the state set. $f(c_i)$ is the number of partial invalid states of cycle c_i corresponding to the state set the test generator traverses. A new testability measure is proposed based on the invalid states. Scan flip flop selection is to meet the unsatisfiable signal requirements.

$$T(n) = \sum_{c_i} f(c_i) \quad (5)$$

The TIP measure of each flip flop n is,

$$TIP(n) = \sum_{c_i} f(c_i) \cdot k_i \quad (6)$$

where k_i is the size of cycle c_i, c_i represents all the cycles containing flip flop n.

Algorithm 3 (scan flip flop selection based on the states the test generator traverses)

1. For each cycle c_i, $f(c_i) \leftarrow 0$;
2. Record the states in S of the test generator traversed during the initial stage;
3. For each state Q_i in S, map state Q_i to the cycles; if there is a new partial valid state for each cycle c_i, $f(c_i) \leftarrow f(c_i) + 1$;
4. Assign a testability measure to each flip flop using equation (5); assign a TIP measure to each flip flop using equation (6).

5. Choose the flip flop n with the greatest TIP measure as the scan flip flop. For each cycle c_i, c_i contains n; the testability measure of each other flip flop n_1 in c_i is decreased by $f(c_i)$, and the TIP measure is decreased by $f(c_i) \cdot k_i$.

6. If no more scan flip flops are needed, end the algorithm; otherwise, go to step 5.

Tests are generated for the fully scanned circuit. State information is obtained from the pseudo-primary inputs corresponding to the test set of the fully scanned circuit. These states indicate signal requirements of the circuit to the flip flops.

Algorithm 4 (states from the tests for the combinational logic)

1. For each cycle c_i in the original circuit, $f(c_i) \leftarrow 0$; generate tests for the fully scanned circuit;

2. Obtain state set S from the PPI part of each test vector. Map each state obtained above to each cycle c_i. If c_i gets a new partial state, $f(c_i) \leftarrow f(c_i) + 1$;

3. Estimate the testability measure of each flip flop using equation (5). Estimate the TIP measure of each flip flop using equation (6);

4. Choose the flip flop n with the largest TIP measure as a scan flip flop. Set the testability measure and the TIP measure of n as 0. For each cycle c_i, c_i contains n, decrease the testability measure of n by $f(c_i)$, and decrease the TIP measures of n by $k_i \cdot f(c_i)$;

5. If no more scan flip flops are required, end the algorithm; otherwise, go to 4.

6 Experimental Results

For simplicity, we shall call the valid state-based algorithm *scan1*, the test generation states-based algorithm *scan2* and the fully scanned circuit test-based algorithm *scan3* respectively. Table 3 presents the HITEC test generation results for the three partial scan designs. In the table, *scan* represents the number of scan flip flops. *cov.i*, *TEi*, *cpui* ($i \in \{1, 2, 3\}$) represent fault coverage, test efficiency and CPU time of the three algorithms respectively. In comparison with the three algorithms, *scan1* demonstrates the best performance, *scan2* the second best, and *scan3* the worst.

Table 3 shows the number of scan flip flops (*mfvs*) selected by pscan [5]. pscan [5] selects the minimum number of scan flip flops needed to break all the cycles. Fewer scan flip flops are selected for almost all the listed circuits than by pscan [5].

Table 4: Comparison with Opus

Cir.	scan	FC	TE
s298	1/1	0.948/0.948	1.00/1.00
s344	5/5	0.990/0.994	1.00/1.00
s349	5/5	0.986/0.983	1.00/0.997
s382	9/9	0.987/0.987	1.00/1.00
s386	5/5	1.00/1.00	1.00/1.00
s400	9/9	0.974/0.971	1.00/1.00
s444	9/9	0.964/0.960	1.00/1.00
s510	5/5	1.00/1.00	1.00/1.00
s526	3/3	0.755/0.755	0.77/0.77
s641	7/7	0.974/0.946	1.00/1.00
s713	7/7	0.912/0.885	1.00/1.00
s820	2/4	0.998/0.998	0.998/0.998
s832	2/4	0.984/0.984	1.00/1.00
s953	3/5	1.00/1.00	1.00/1.00
s1488	2/5	1.00/1.00	1.00/1.00
s1494	3/5	0.992/0.992	1.00/1.00
s5378	30/30	0.936/0.936	0.962/0.962
s35932	150/306	0.898/0.898	1.00/1.00

Table 4 compares the experimental results with Opus [7]. In item a/b, a and b indicate the parameters of *scan1* and Opus respectively. *scan1* obtains better results than Opus for almost all the circuits.

7 Conclusions

We found it is unnecessary to break all the cycles as in the conventional structure-based partial scan design methods. The circuit state information is obtained via logic simulation, which is used to guide scan flip flop selection. Depending on the obtained state information, some cycles in the circuit assume only a few valid states, while for others most partial states are valid. The cycles with few states are typically the cause of test generation complexity. A measure based on the density of encoding was used to direct partial scan flip flop selection. States traversed by the test generator in the initial stage were used to select scan flip flops. Finally, circuit state information was also obtained by generating tests for the fully scanned circuit. This circuit state information provides the signal requirements of the circuit for the flip flops. Scan flip flops were selected to meet these signal requirements. Experimental results demonstrate all three schemes are effective.

Table 3: Comparison of the Three Schemes

Circuits	FFs	SFF	mvfv	**scan1**			**scan2**			**scan3**		
				cov.1	TE1	cpu1	cov.2	TE2	cpu2	cov.3	TE3	cpu3
s298	14	1	1	0.948	1.00	14.0	0.948	1.00	13.9	0.948	1.00	13.8
s344	15	4	5	0.988	1.00	26.0	0.988	1.00	3.0	0.988	1.00	25.2
s349	15	4	5	0.983	1.00	1.9	0.983	1.00	1.7	0.983	1.00	2.00
s382	21	6	9	0.982	1.00	4.5	0.98	1.00	6.8	0.98	1.00	6.7
s386	6	3	5	0.979	1.00	6.2	0.948	1.00	3.0	0.974	1.00	1.9
s400	21	5	9	0.958	1.00	10.4	0.972	1.00	7.7	0.942	1.00	10.6
s444	21	5	9	0.949	1.00	20.0	0.949	1.00	11.5	0.945	1.00	11.6
s526	21	3	3	0.755	0.77	316.4	0.755	0.77	316	0.755	0.77	316
s641	21	6	7	0.946	1.00	4.6	0.946	1.00	4.6	0.942	1.00	5.7
s713	19	6	7	0.881	1.00	11.8	0.881	1.00	12.0	0.881	1.00	12.0
s820	5	2	4	1.00	1.00	6.00	1.00	1.00	6.0	1.00	1.00	12.3
s832	5	2	4	0.984	1.00	5.0	0.984	1.00	4.8	0.984	1.00	8.7
s953	29	3	5	1.00	1.00	3.7	1.00	1.00	3.7	1.00	1.00	9.6
s1423	74	20	21	0.831	0.84	597	0.86	0.87	468	0.81	0.82	673
s1488	6	3	5	1.00	1.00	12.6	1.00	1.00	21.6	1.00	1.00	22.1
s1494	6	4	5	0.992	1.00	8.9	0.992	1.00	35.8	0.992	1.00	40.1
s5378	179	30	30	0.936	0.962	597	0.936	0.962	439	0.936	0.962	446.9
s35932	1728	150	306	0.898	1.00	4110	NA	NA	NA	0.898	1.00	4272

References

[1] T. E. Marchok, A. E. Maleh, W. Maly, and J. Rajski, "Complexity of Sequential ATPG," in the Proc. of *European Design & Test Conf.*, pp. 252-261, 1995.

[2] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[3] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," in the Proc. of *European Conf. on Design Automation*, pp. 214-218, 1991.

[4] K. T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. Comput.*, Vol. 39, No. 4, pp. 544-548, 1990.

[5] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip Flops," in the Proc. of *ACM/IEEE Design Automation Conf.*, pp. 81-86, 1994.

[6] T. H. Chen and M. A. Breuer, "Automatic Design for Testability via Testability Measures," *IEEE Trans. CAD*, vol. CAD-4, pp. 3-11, 1985.

[7] V. Chickermane and J. H. Patel, "An Optimization Based Approach to The Partial Scan Design Problem," in the Proc. of *IEEE Int. Test Conf.*, pp. 377-386, 1990.

[8] D. H. Lee and S. M. Reddy, "On Determining Scan Flip Flops in Partial Scan Designs," in the Proc. of *IEEE Int. Conf. Computer-Aided Design*, pp. 322-325, 1990.

[9] P. S. Parikh and M. Abramovici, "A Cost Based Approach to Partial Scan," in the Proc. of *ACM/IEEE Design Automation Conference*, pp. 255-259, 1993.

[10] I. Pomeranz and S. M. Reddy, "LOCSTEP: A Logic Simulation Based Test Generation Procedure," in the Proc. of *IEEE Fault-Tolerant Computing Symposium*, pp. 110-118, 1995.

[11] A. Lioy, P. L. Montessoro, and S. Gai, "A Complexity Analysis of Sequential ATPG," in the Proc. of *Int. Symposium on Circuits and Systems*, pp. 1946-1949, 1989.

[12] N. Jiang, R. M. Chou, and K. K. Saluja, "Synthesizing Finite State Machines for Minimum Length Synchronizing Sequence Using Partial Scan," in the Proc. of *IEEE Fault-Tolerant Computing Symposium*, pp. 41-49, 1995.

[13] V. Boppana, I. Hartanto, and W. Kent Fuchs, "Fault Diagnosis Using State Information," to appear in the Proc. of *IEEE Fault-Tolerant Computing Symposium*, 1996.