Partial Scan beyond Cycle Cutting *

Gurjeet S. Saund, Michael S. Hsiao and Janak H. Patel
Center for Reliable and High-Performance Computing
University of Illinois at Urbana-Champaign, Urbana, IL 61801
http://www.crhc.uiuc.edu/

Abstract

This paper addresses the problem of flip flop selection for partial scan in sequential circuits. In particular it addresses some of the shortcomings of the popular flip flop selection methods, based on cutting cycles present in the graph of the circuit structure. Previous approaches assume that cutting all cycles makes the circuit totally testable, which is not always true. In the proposed approach, first subsets of flip flops are formed based on cycles in the S-graph and flip flops with self-loops. Flip flops are selected from these subsets based on a testability measure which uses an approximate valid state analysis. Once a flip flop is selected from a subset, testability measures may indicate the need for more flip flops, thus possibly selecting more flip flops than required for minimum cycle cutting. The goal is to select the fewest number of flip flops required to obtain high fault coverage for all partial scan circuits. Experimental results on the benchmark circuits show that a test generation efficiency near 100% is achieved for most circuits.

1 Introduction

Partial Scan Design is a design for testability(DFT) method which has been proposed to improve the testability of sequential circuits. It is an attractive DFT solution compared to the full scan approach, since it has low area and performance overhead. It allows a designer to trade off test generation complexity and fault coverage, with area and delay overheads. The area overhead incurred using partial scan is the area of additional logic in the scan latches and routing of the scan chain. The overhead can be approximated by the number of flip flops included in the scan chain. The goal in this paper, is to provide the highest fault coverage for a circuit, by scanning the fewest number of flip flops. Fewer flip flops in the scan chain implies a lower area overhead.

*This research was supported in part by the Semiconductor Research Corporation under contract SRC 96-DP-109, in part by DARPA under contract DABT63-95-C-0069, and by Hewlett-Packard under an equipment grant.

1.1 Previous Work

Previous research on selection of flip flops for partial scan, can be classified into three main categories: testability measure based, test generation based, and structure based methods. Additional methods have been layout based, re-timing based, targeted at sharing functional logic in the scan chain for reduced overhead, or test time reduction by reordering the scan chain.

Testability measures have been used as the first measures [1] to guide flip flop selection. The primary problem associated with testability measure based methods, is that the fault coverage obtained for complex circuits may not be high. Another shortcoming of simple measures [2, 3] is that they may not correlate with test generator complexity, hence requiring more scan flip flops to achieve desired coverage. More complex measures based on test generator execution times [4] try to avoid the above, but can be computationally expensive.

Test generation based methods use information from a test generator to aid in the selection of flip-flops. Methods which use an initial pass of the test generator to drop easy to detect faults [5, 6] would be impractical for large circuits. Other methods of targeting specific faults [7] using a test generator can be computationally expensive. Unjustified states and unpropagated fault effects have also been used for selection of flip flops [6, 7]. Another method [8], which combines test generation, structure and testability based approaches can be expensive for large circuits. This method does not take into consideration the effect of scanning a flip flop in the circuit, on subsequent selections.

Structure based methods have been more widespread, since test generation complexity is considered to be exponential in the length and number of cycles in a circuit and linear in its sequential depth [9]. Subsequently, the idea of cutting cycles in a circuit, ignoring the self loops, has been mapped into a graph theoretic problem of finding the minimum feedback vertex set. A number of papers [10, 11, 12, 13] have proposed algorithms to find a near minimal or

0731-3071/97 $10.00 © 1997 IEEE

the minimal subset of flip flops to cut all cycles. However, according to [5] it may not be necessary to cut all cycles in a circuit to test some circuits fully. This idea has been used for scan selection [14, 15], to selectively cut cycles based on reachable states. However, the testability measure these methods were based on is greatly inaccurate for circuits with large cycles. If just one flip flop is scanned from a large cycle, the cycle is declared as having all partial states reachable, which is not always true. Secondly, cutting all cycles does not detect all faults for some circuits [5]. In this paper, we give results of selection for circuits having large cycles. We also address the problem of low coverage in circuits with no cycles.

We use valid state analysis to evaluate the testability of the circuit. This is based on the work by Marchok et al. [16], who identified that density of encoding can be used as one of the measures for evaluating test generation complexity. The density of encoding (ED) can be expressed as

$$ED = \frac{V}{2^n} \quad (1)$$

where V is the number of valid states, and n is the number of flip flops in the circuit. The density of encoding for circuits with a large number of flip flops is typically far less than one.

Structure based methods theorize that test generation complexity is exponentially related to the size and number of cycles. If a cycle of length k has 2^k valid states, then it would not cause many backtracks in the test generation process. If another cycle had just one valid partial state, then it would lead to numerous backtracks. Hence one should cut the latter and not the former cycle. Selective cycle cutting is performed in this manner. However, if the number of states in the cut cycle does not increase significantly, another flip flop from the remaining subset may need to be scanned. Thus, flip flops from cut cycles should not be dropped from further consideration.

1.2 Contribution

We propose a technique of scanning flip flops using valid states analysis of flip flop subsets in the circuit. Previous methods [14, 15] assume that cutting a cycle makes the cycle totally testable. This assumption can be inaccurate for large and complex circuits, where several flip flops may have to be selected from just one cycle to make it testable. In such complex circuits, *minimal* cycle cutting will not be sufficient to obtain high fault coverage.

The proposed approach differs from previous methods in that, the cycles are not the primary focus of the testability analysis. Our emphasis is on

- Coding density of a subset of flip flops, which may be part of a cycle or a collection of flip flops with self loops
- Scaling the approach to large circuits.

The method tackles the problem of handling large subsets, by breaking them into subsets of smaller size. Performing the testability analysis iteratively allows us to get a more accurate selection of flip flops.

This paper is divided into the following sections: Section 2 covers some of the previous work and some definitions; Section 3 describes the flow of the selection process; Section 3.1 describes how logic simulation is used to obtain the valid states in a circuit; Section 3.2 describes how subsets are formed, Section 3.3 describes the testability measures used for flip flop selection; Section 4 compares our method with previous approaches; and we conclude in Section 5.

2 Background

We cover some definitions and notations which are used in the remaining part of the paper. Some of the following definitions have been taken from work by D. Xiang et al. [14].

Definition 1: A **state** is an assignment of boolean values { 0, 1 } to the outputs of flip flops. The **reset state** is a state which is reachable from every other state.

Definition 2: A state is called a **valid state** if it is reachable from a reset state; a state is called an **invalid state** if it is not reachable from a reset state.

A valid state can be justified using only the primary inputs of a circuit, whereas justification of invalid states leads to backtracks in the process of test generation. Logic simulation with pseudo-random inputs is used to generate valid states for a circuit.

A cycle of length k has 2^k possible states. If each one of these states in the cycle is valid, very few backtracks would be needed. Hence, if one were to make all partial states valid for every subset, the number of backtracks would be significantly reduced.

Definition 3: The vertices of a **S-graph** of a sequential circuit, are the flip flops of the circuit. There exists an edge from vertex u to vertex v in the S-graph if there exists a combinational path from flip flop u to flip flop v in the circuit.

If an edge starts and ends at the same vertex v then it is called a **self loop**. Sub-circuits containing flip flops with self loops can increase the test generation complexity significantly. In some cases test sequence lengths can be exponential in the number of flip flops with self loops (e.g. counters). Practical circuits contain numerous flip flops with self loops. In cycle cutting

based schemes, flip flops with self loops are ignored because cutting all cycles would require including all such flip flops in the scan chain. We identify sub-circuits with flip flops having self loops, which are not part of any cycle, and consider them for scan selection.

Definition 4: A **strongly connected component** of a directed graph is a subgraph, such that any two vertices in the subgraph have a cycle in common. The strongly connected components of a graph can be found in linear time using a depth first search based algorithm [17].

Definition 5: Assume a **state** is an n-tuple of signal values $(v_1, v_2, \ldots, v_n)$ where n is the number of flip flops in the circuit and a **partial state** is a k-tuple of signal values $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$, where $i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, n\}$ and $k < n$. A partial state is called a **valid partial state** if there exists a valid state which contains the partial state. A partial state is called an **invalid partial state**, if no valid state contains the partial state. A **State Mapping** maps a state to all subsets in the circuit.

Definition 6: Density of Encoding [16] 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.

3 Methodology

The methodology followed by our approach selects the fewest number of scan flip flops required to obtain the highest fault coverage and close to complete test generator efficiency (near 100%). We use a testability measure based on valid state analysis of flip flop subsets to determine scan flip flops. Flip flop subsets are formed based on cycles and flip flops with self loops in the S-graph of the circuit. Logic Simulation is performed to identify the valid states in the circuit. State mapping is then performed based on the states which have been reached by the simulator (as described in Section 2). The testability measure for each subset is updated, and the potential improvement in testability obtained by scanning each flip flop is computed based on the states reached during logic simulation. The flip flop which would provide the maximum improvement in testability is selected for inclusion in the scan chain. After a flip flop is selected, it is dropped from all subsets to which it belongs. The output of the scanned flip flop is considered as a primary input, and it's input as a primary output for the purpose of logic simulation and subsequent test generation.

The sequence of logic simulation and scan selection is repeated until *all* partial states are valid for each subset, or if the user defined limit on the number of scan flip flops is reached. More than one flip flop may be scanned in each iteration, using a dynamic heuristic to determine the number of flip flops to scan. This iterative loop, as illustrated in Figure 1, has the advantage of being able to stop scanning flip flops when the testability of the circuit cannot be improved on further. This provides the designer with an upper bound on the number of flip flops to include in the scan chain. A number of scan selection methods require the designer to give an upper bound on the number of flip flops to scan as a parameter. These methods are impractical as they require the designer to perform a number of iterations of the test generator and the scan tool in an attempt to come up with a sufficient number of flip flops required for high coverage.

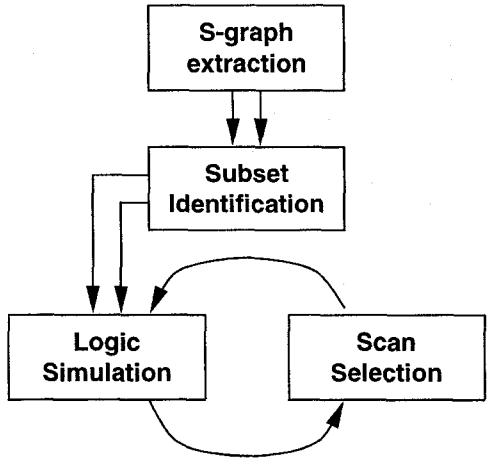

Figure 1: Partial Scan methodology

The remaining part of this section describes the enumeration of valid states using logic simulation, the method of forming subsets, and the testability measures.

3.1 Enumeration of Valid States

Logic Simulation is used to estimate the number of valid states of a circuit as in [14]. The valid states of a machine can be easily estimated by logic simulation using random input vectors. If the synchronizing sequence or reset state is given, we start the simulation from the reset state of the circuit. Otherwise, all flip flop values are considered to be unknown at the beginning of the simulation.

We use bit parallel logic simulation to find valid states of the circuit. Since we can start the simulation from 32 different initial states, we expect to reach more valid states than starting from one initial state. In subsequent iterations, the states of the circuit at the end of the previous iteration are used to start the simulation.

The maximum number of new states a circuit can reach if the circuit is simulated for T time-frames is

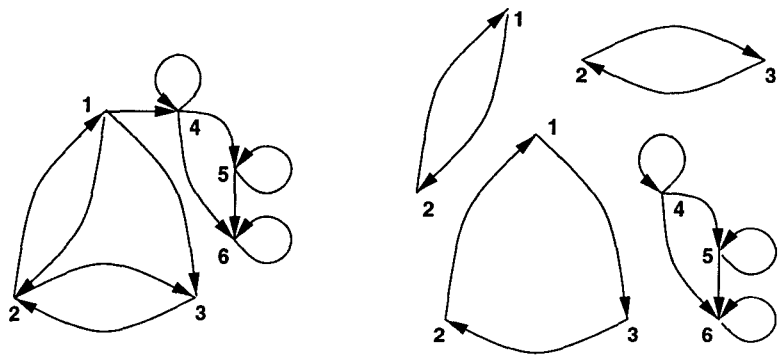

Figure 2: Subset Formation in a circuit

$32 \cdot T$. Logic simulation is done for 200 time-frames within each iteration. Hence, we can traverse a maximum of 6400 new states between iterations. For a number of circuits for which the valid state space is very large, or the flip flops are many in number, we will not reach all valid states. The states which are reached using logic simulation are *easily* justifiable using a test generator. States which are hard to justify, add to test generation time and complexity. Since we do not reach hard to justify states, the states reached correlate to the states reachable using a test generator. However there are many states which a test generator can reach, but are not reachable using a pseudo-random logic simulator. Unlike cycle cutting algorithms we do not cut all cycles, particularly cycles with a large number of valid states. In addition, the number of valid states used is only a heuristic for selecting the scan flip flops and therefore any reasonable estimates on their measure are adequate for our purpose.

3.2 Subset Formation

Subsets are formed based on cycles and flip flops with self loops in the circuit. First, the S-graph of the circuit is extracted from the net-list in linear time, and subsequently strongly connected components of the S-graph are identified [17]. An approximate algorithm is used to enumerate cycles (ignoring self loops) within each strongly connected component. Each enumerated cycle forms a subset. Note that the subsets are not required to be disjoint. Alternately the subsets could be determined based on fan-in cones of each flip flop. This approach has not been studied in this paper.

It has been observed that counters and flip flops with self loops can significantly increase test generation complexity [10]. We also form subsets which include flip flops with self loops. A flip flip with a self loop which does not belong to any subset forms a new subset. A flip flop with a self loop is added to a subset if an adjacent flip flop with a self loop is already in that subset. These subsets can be formed in linear time. In Figure 2, subsets $\{1, 2, 3\}$, $\{1, 2\}$, $\{2, 3\}$, are obtained using cycle enumeration, and $\{4, 5, 6\}$ is obtained using the adjacency method targeting flip flops with self loops.

The total number of possible valid states for a subset of size 40 is of the order of 10^{12}. Cycles of this size are numerous in the large ISCAS benchmark circuits. Reaching these many states within one iteration is too costly. In our method, where a maximum of 6400 states can be reached in an iteration, large subsets can incorrectly bias our selection. If all partial states for such a large subset were valid, verifying this using logic simulation is too expensive. Hence to make the problem tractable for large circuits, we break large subsets into smaller sizes such that the root mean square deviation of the new subset sizes from a target size(12 - determined empirically) is minimum. In some circuits, all flip flops of a large subset are hard to control and are required to activate a fault or propagate a fault effect. Our heuristic of breaking large subsets may declare such a circuit to be fully testable prematurely. All partial states may be declared valid only for broken parts of the large subset, but not the entire subset. This can be remedied by increasing the target subset size for broken subsets.

3.3 Partial Scan based on Valid States

We use a testability measure based on valid states to evaluate the testability of flip flop subsets in a circuit. The state information obtained using logic simulation described in Section 3.1 is used to select the flip flops for inclusion in the scan chain.

As stated before in Section 1.1 if a subset has a high density of encoding, it would lead to a fewer number of backtracks during ATG than a subset with a lower density of encoding. If there are a number of cycles and

self loop based circuits with all possible partial states being valid, then the testability of the circuit should be good. Each state reached in the logic simulation is mapped to the subsets in the circuit as described in Section 2. The testability measure for each flip flop f is evaluated as:

$$T(f) = \sum_{\forall i,\ s.t.\ f \in S_i} T(S_i) \tag{2}$$

$$T(S_i) = \begin{cases} \frac{2^{k_i}}{f(S_i)} - 1 & \text{if } k_i \leq 12 \\ (k_i - 11) \cdot \frac{2^{12}}{f(S_i)} - 1 & \text{if } k_i > 12 \\ 0 & \text{if } T(S_i) < 0 \end{cases} \tag{3}$$

where 12 is an empirical constant, $f(S_i)$ is the number of valid partial states obtained for subset S_i obtained through logic simulation, and k_i is the size of the subset.

The testability measure reflects the possibility of entering invalid states in the process of test generation. The measure is biased towards subsets of size less than 12. We reduce the effect of the large subsets, because the number of partial states reachable in logic simulation are far fewer than the possible valid states of the subset. In test generation terms, if a significant number of partial states are valid for a pseudo-random input logic simulator, the test generator can backtrack to one of these states and easily justify it.

The testability improvement potential (TIP) measure for each node is used to evaluate potential testability improvement of the circuit if the node is chosen to be the scan flip flop. The TIP measure for each flip flop is:

$$TIP(f) = \sum_{\forall i,\ s.t.\ f \in Si} T(S_i) \cdot k_i \tag{4}$$

In each iteration, the node with the largest TIP measure is chosen as a scan flip flop. More than one flip flop may be selected in one iteration, to speed up the selection process. A dynamic heuristic is used to determine the number of flip flops to scan in each iteration. This measure takes into account the number of non-trivial strongly connected components(i.e. size greater than one), self loop based subsets as described and their respective sizes. Since cycles can only exist within SCC's, several flip flops with the largest TIP measure in each SCC can be selected. However, it could happen that small SCC's do not reach all valid states, due to the limitations of logic simulation. To avoid scanning flip flops from SCC's or acyclic sub-circuits the following heuristic is followed.

$$Num\_Scan = \lceil 0.1 \cdot Largest\_SCC \rceil$$

$$Max\_from\_one\_SCC = \left\lceil \frac{Num\_Scan}{Num\_Large\_SCC} \right\rceil$$

In each iteration we scan up to 10% of the size of the largest SCC or acyclic subset at a time. The parameter $Max\_from\_one\_SCC$ ensures that the selection is distributed across all SCC's. The dynamic testability measure ensures that the selection of flip flops within an SCC is intelligent. Num_Large_SCC refers to SCC's or acyclic subsets which have more than 8 elements and does not have all partial states valid.

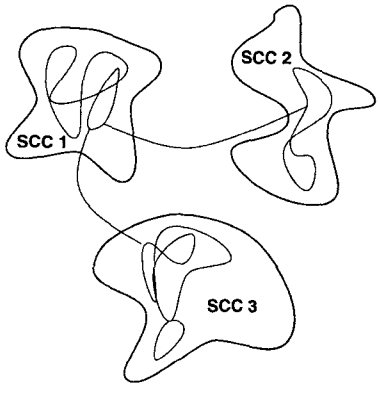

Figure 3: SCC's in an S-graph

Example 1: Let three SCC's shown in Figure 3 represent a circuit, where the sizes of each SCC are 20, 10 and 5 respectively. Num_Large_SCC in this case would be 2. The maximum number of flip flops scanned in the first iteration would be 2 and the maximum number of flip flops scanned from each SCC would be $\left\lceil \frac{Num\_Scan}{Num\_Large\_SCC} \right\rceil = \lceil \frac{2}{2} \rceil = 1$.

The following procedure describes the steps involved in scanning flip flops using valid state analysis. The conditions for selecting a limited number of flip flops from the same strongly connected component is omitted for clarity.

Procedure 1:

1. For each subset S_i, Number of valid partial states $f(S_i) \leftarrow 0$;
2. Perform logic simulation;
3. For each state Q_i in the states reached, do
   (a) Perform state mapping of Q_i to partial state $Q_i^{'}$ for each subset S_i;
   (b) if $Q_i^{'}$ is a new partial state for S_i, $f(S_i) \leftarrow f(S_i) + 1$;
4. For each subset S_i, calculate the testability of the subset using equation 3.
5. For each flip flop f, calculate the TIP measure using equation 4;

6. If the TIP measure for each flip flop is 0, then end the Procedure;
7. Choose the flip flop f_s with the largest TIP as a scan flip flop.;
8. For each subset S_i, such that $f_s \in S_i$;
   (a) Decrease the TIP measure for each node in the subset by $k_i \cdot T(S_i)$;
   (b) $S_i \leftarrow [S_i - \{f_s\}]$;
9. If flip flops scanned in one iteration are fewer than the maximum number to scan in an iteration then goto 7;
10. If the number of flip flops scanned are fewer than the maximum number to scan then goto 1, otherwise end the Procedure;

Example 2: Consider the scan selection based on valid state analysis of a circuit with 6 flip flops. Assume that the S-graph of the circuit is as shown in Figure 2a) and that the valid state set consists of 100000, 101001, 110010, 111110 . Figure 2b) shows the subsets formed from the S-graph. According to the 4 valid state set, the number of partial valid states for subsets {1, 2, 3}, {1, 2}, {2, 3}, {4, 5, 6} are 4, 2, 4, and 4 respectively. Testability measures for flip flops $1-6$ computed using equation 2 are:

$$T(1) = \frac{8}{4} - 1 + \frac{4}{2} - 1 = 2$$

$$T(2) = \frac{8}{4} - 1 + \frac{4}{2} - 1 + \frac{4}{4} - 1 = 2$$

$$T(3) = \frac{8}{4} - 1 + \frac{4}{4} - 1 = 1$$

$$T(4) = T(5) = T(6) = \frac{8}{4} - 1 = 1$$

TIP measures for nodes 1-6 computed using equation 3 are as follows:

$$TIP(1) = 1 * 3 + 1 * 2 = 5$$

$$TIP(2) = 1 * 3 + 1 * 2 + 0 * 2 = 5$$

$$TIP(3) = 1 * 3 + 0 * 2 = 3$$

$$TIP(4) = TIP(5) = TIP(6) = 1 * 3 = 3$$

Either flip flop 1 or 2 is selected based on their connectivities. Flip flops with the same TIP measures are selected by comparing the number of fan-outs first, and then the number of fan-ins of each of the flip flops. Assuming that flip flop 2 is selected, the subsets are then modified to {1, 3}, {1}, {3}, {4, 5, 6}. Logic simulation is then performed to observe the effect of scanning flip flop 2, and TIP measures for the remaining flip flops are recomputed in the next iteration.

4 Results

In this section we show how changing the target broken size affects the selection, and compare the performance of our scan selection method versus Opus [12]. The scan selection tool, **ZScan** was implemented in C. Experiments are performed on the ISCAS89, ISCAS93 sequential benchmark circuits and some synthesized circuits. A brief description of the synthesized circuits is as follows: *am2910* is a 12-bit microprogram sequencer; *mult16* is a 16-bit 2's complement shift-and-add multiplier; *div16* is a 16-bit divider which performs division by repeated subtractions; *gcd* is a circuit which finds the greatest common divisor of two 16-bit numbers; and *barcode* is a barcode reader. *gcd* and *barcode* are synthesized versions of the circuits given in the HLSynth95 benchmark suite.

Table 1 illustrates the loss of information in breaking large subsets into smaller sized subsets. In this experiment, HITEC was run for a time limit of 5 seconds with a backtrack limit of 100000 per fault. Logic simulation was performed for 300 time steps, in each iteration. **Unscanned** refers to the original circuit, **Not Broken** to the scanned circuit, when subsets are not broken. The other two columns refer to the scanned circuits when the target subset size(Tgt Sbst Sz) is as shown. #FF refers to the number of flip flops in the circuit, #SFF is the number of flip flops which are scanned, FC refers to the Fault Coverage ($FC = \frac{Detected\ Faults}{Total\ Faults}$) and TE is the Test Generator Efficiency ($TE = \frac{Detected\ Faults+Untestable\ Faults}{Total\ Faults}$) of the generated test vectors. When the target subset size for the broken cycles is small, then the selection method can prematurely declare the circuit to be fully testable. In *s420.1*, there is an acyclic sub-circuit of 16 flip flops with self loops. For a target subset size of 10 - the acyclic circuit is broken into two subsets of size 8 each, thus introducing inaccuracy. For *s838.1*, the loss of accuracy is observed across the unbroken subsets versus both of the broken subsets selection. On reducing the target broken subset size, the number of flip flops selected are fewer and the testability obtained is also lower.

Table 2 compares the Original circuit, which refers to the unscanned circuit, with the scanned circuit obtained using Opus, and the scanned circuit obtained using ZScan. HITEC is run for a up to 50s per fault and a backtrack limit of 1,000,000 for the smaller circuits. Circuits marked with a (*) in the table, are run for 5 seconds per fault and a backtrack limit of 100,000. Our experiments were performed on a HP9000-J200 workstation with 256MB of memory.

Opus uses heuristics to find a close to minimal subset of flip flops for cutting all cycles in a circuit. After

Table 1: Loss of information due to breaking large subsets

Circuit	Unscanned		Not Broken		Tgt Sbst Sz = 12		Tgt Sbst Sz = 10	
	#FF	FC(TE)	#SFF	FC(TE)	#SFF	FC(TE)	#SFF	FC(TE)
s420.1	16	41.3(99.7)	16	100(100)	15	95.6(100)	9	28.4(92.1)
s838.1	32	29.6(83.2)	32	100(100)	29	53.7(99.4)	29	53.7(99.4)

cutting all cycles it uses a SCOAP based measure to scan additional flip flops. To make a fair comparison with Opus, we scan the maximum of the number of flip flops required for complete cycle cutting or the number of flip flops scanned by ZScan. The time for scanning using Opus is at most 830 seconds for s38417, being lesser than 5 seconds for most circuits.

We compare the two selections by first comparing the test generator efficiency. If the test generator efficiency is 100%, then we compare the number of flip flops scanned. If the test generator efficiencies are the same but lower than 100%, then the fault coverages are compared second, followed by the number of flip flops scanned. The results in **bold** indicate better performance. The parameters used for ZScan are as follows: logic simulation is performed for 200 time steps between iterations; large subsets are broken into smaller sizes(target size being 12); the dynamic heuristic is used to find the number of flip flops to scan within each iteration; the user limit given for the flip flops is the maximum possible(i.e. the number of flip flops in the circuit). s1196, s1238 from ISCAS89 and s1196.1 from ISCAS93 have been excluded since no flip flops are scanned from these circuits.

For circuits s298, s526 and s4863 a number of flip flops are scanned from the acyclic part without improving the coverage significantly. Even if we were to avoid scanning acyclic subsets in the circuits, we can obtain 100% test generator efficiency for these circuits. There is a self loop based subset of size 11 in s298, which does not have many states reachable during logic simulation. Nine elements from this subset are selected from scan even though it does not contribute to test generation complexity. The flip flops in this subset are loosely connected to each other. This indicates that the heuristic used to form acyclic subsets can be made sophisticated to target different the kind of structures and avoid others. It also indicates that the states reachable using logic simulation are an approximate indicator of test generator efficiency.

The test generator efficiency reported for most of the circuits is near 100%. 25 of the 43 benchmark circuits with ZScan give 100% test generator efficiency; 11 scanned circuits give greater than 99% efficiency; with all other circuits except s3330, s3384 and s35854 having an test generator efficiency of 95% or greater. Though these three circuits do not achieve high test efficiency for ZScan, the testability is similar to Opus. This indicates that the number of flip flops scanned is far fewer than those required for high coverage. This can be because of breaking large subsets into smaller sizes. We also may have to target structures other than cycles and self loop based sub-circuits to achieve higher coverage.

From the ISCAS89 benchmarks, which are listed first in Table 2, ZScan performs better than Opus for 20 out of 25 circuits; has comparable results for 6 circuits and performs worse than Opus for s15850.1. ZScan obtains slightly less coverage for these circuits, but it provides an end point which tells the tool to stop scanning. Complete cycle cutting in Opus scans fewer number of flip flops and obtains much lower fault coverage and test generator efficiency for s1423, s5378, s9234, s13207.1, s15850.1 and s38417.

For the ISCAS93 benchmarks, ZScan performs better for 6 out of 12 circuits, gives comparable results for 3 circuits, and is worse for 3 circuits when compared with Opus. s991 has a flip flop with a self loop which is not affected by any other node in the S-graph. This flip flop just toggles on the application of a clock, however during test generation this flip flop is uninitializable. Opus gives extremely low coverage for this circuit because it ignores such structures.

For the synthesized circuits, ZScan performs better for 1 of the 5 circuits, gives comparable results for another circuit, and performs worse for 3 other circuits. The S-graph of *gcd* contains a clique of size 33, and enumerating all the cycles in the clique is exponential in complexity. The approximate cycle enumeration, and the breaking of subsets introduces some inaccuracy in the selection.

The selection times for most circuits are small except for some of the larger circuits. These times however are a small fraction of the total test generation time required for the original circuits, which can be of the order of tens of hours without providing a high coverage.

We also compare our results with Opscan [15]. Opscan does not provide results for large circuits in both the ISCAS89 and ISCAS93 benchmark suites, since it

is not scalable to large circuits. For the table reporting test generation results, we obtain higher test generator efficiency in 5 circuits, lower efficiency in 1 circuit and the same in all other 23 circuits reported. The aggregate number of flip flops scanned (except s35932 because it is misleading, since it is fully testable prior to scan) is 275 versus 230 scanned by Opscan. The selection times (except s35932) for both methods are also comparable. This illustrates that in-spite of using subsets as a basis for selection (i.e. not declaring cycles to be fully testable when a cycle is broken) we do not scan significantly more flip flops in the smaller circuits. For s5378, we scan just 48 flip flops compared to 80 for Opscan to obtain the same test generator efficiency.

5 Conclusions

In this paper, we have introduce an approach to increase the low fault coverage and test generation efficiency of large circuits scanned using cycle cutting based approaches. Our approach targets two limitations of previous schemes: cycles which have been cut, and sub-circuits consisting of flip flops with self loops. Flip flop subsets are formed based on cycles and flip flops with self loops in a circuit. Subsequently, a testability measure based on valid states is used to select scan flip flops from these subsets. Logic simulation is used to estimate the number of valid states for the circuit. An iterative procedure performs logic simulation and scan selection, until the testability of the circuit cannot be improved. An algorithm, which finds the number of flip flops to scan in an iteration, is given to speed up the selection of scan flip flops. Experimental results show that the approach is scalable to large circuits. For small circuits, 100% test generation efficiency is reached by scanning fewer flip flops than those required for minimal cycle cutting. A test generator efficiency of near 100% is achieved for most other benchmark circuits within reasonable time.

References

[1] E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology," in *Proceedings of the International Test Conference*, pp. 153–162, November 1980.

[2] L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits," *IEEE Transactions on Circuits and Systems*, vol. 26, pp. 685–693, 1979.

[3] P. S. Parikh and M. Abramovici, "Testability-Based Partial Scan Analysis," *Journal of Electronic Testing*, vol. 7, pp. 225–259, August/October 1995.

[4] K. S. Kim and C. R. Kime, "Partial Scan Flip-Flop Selection by use of Empirical Testability," *Journal of Electronic Testing*, vol. 7, pp. 47–60, August/October 1995.

[5] V. Chickermane and J. H. Patel, "An Fault Oriented Partial Scan Design Approach," in *Proceedings of the International Conference on Computer Aided Design*, pp. 400–403, November 1991.

[6] V. Boppana and W. K. Fuchs, "Partial Scan Based on State Transition Modeling," in *Proceedings of the International Test Conference*, pp. 538–547, October 1996.

[7] I. Park, D. S. Ha, and G. Sim, "A New Method for Partial Scan Design based on Propagation and Justification Requirements of Faults," in *Proceedings of the International Test Conference*, pp. 413–422, 1995.

[8] I. Hartanto, V. Boppana, and W. K. Fuchs, "Identification of Unsettable Flip-Flops for Partial Scan and Faster ATPG," in *Proceedings of the International Conference on Computer Aided Design*, vol. 26, pp. 294–302, November 1996.

[9] K. T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Cicuits with Feedback," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 554–548, 1990.

[10] S. Bhawmik, C. J. Lin, K.-T. Cheng, and V. D. Agrawal, "Pascant: A Partial Scan and Test Generation System," in *Proceedings of the Custom Integrated Circuits Conference*, pp. 17.3.1–17.3.4, May 1991.

[11] D. H. Lee and S. M. Reddy, "On Determining Scan Flip-Flops in Partial Scan Design," in *Digest of the International Conference on Computer Aided Design*, pp. 322–325, 1990.

[12] V. Chickermane and J. H. Patel, "An Optimization based approach to the Partial Scan Problem," in *Proceedings of the International Test Conference*, pp. 377–387, September 1990.

[13] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip-Flops," *Journal of Electronic Testing*, pp. 81–86, August/October 1994.

[14] D. Xiang, S. Venkataraman, W. K. Fuchs, and J. H. Patel, "Partial Scan Design Based on Circuit State Information," in *Proceedings of the Design Automation Conference*, pp. 807–812, June 1996.

[15] D. Xiang and J. H. Patel, "A Global Algorithm for the Partial Scan Design Problem Using Circuit State Information," in *Proceedings of the International Test Conference*, pp. 548–557, October 1996.

[16] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "Complexity of Sequential ATPG," in *The European Design and Test Conference*, pp. 252–261, March 1995.

[17] R. Tarjan, "Depth First Search and Linear Graph Algorithms," *SIAM Journal of Computing*, vol. 1, June 1972.

[18] T. Niermann and J. Patel, "HITEC: A Test Generation Package for Sequential Circuits," in *Proceedings of the European Design Automation Conference*, pp. 214–218, February 1991.

Table 2: Comparison of fault coverage and test generator efficiency for scanned benchmark circuits

Circuit	Original		Opus		ZScan			
	#FF	FC(TE)	#SFF	FC(TE)	#SFF	FC(TE)	Scan Time	ATG Time
s298	14	86.4(94.4)	10	100(100)	10	99.4(100)	8.6s	0.7s
s344	15	95.9(99.1)	5	99.4(100)	5	98.8(100)	7.6s	1s
s349	15	95.6(99.4)	5	98.9(100)	**4**	98.3(100)	5.9s	1s
s382	21	90.9(92.9)	9	98.8(100)	**6**	99.3(100)	8.8s	0.9s
s386	6	81.7(100)	5	100(100)	**4**	100(100)	3.3s	1.2s
s400	21	89.9(93.4)	9	97.4(100)	**5**	97.6(100)	5.4s	3.8s
s420.1	16	41.3(99.7)	16	100(100)	16	100(100)	19.2s	1.3s
s444	21	87.3(92.4)	9	96.0(100)	**6**	96.0(100)	9.2s	1.8s
s510	6	0(100)	5	100(100)	**3**	99.3(100)	2.8s	10.7s
s526	21	65.7(69.0)	15	99.6(100)	15	98.7(100)	17.2s	3.7s
s641	19	86.5(100)	7	94.7(100)	7	99.4(100)	18s	3.7s
s713	19	81.9(100)	7	88.5(100)	7	92.9(100)	13.6s	8.4s
s820	5	95.6(99.4)	4	99.8(99.8)	**2**	**100(100)**	3.2s	8.1s
s832	5	93.9(99.4)	4	98.4(100)	**2**	98.4(100)	3.1s	10.7s
s838.1	32	29.6(83.2)	30	39.5(99.5)	29	53.7(**99.9**)	83s	1900s
s953	29	8.2(100)	5	100(100)	**3**	100(100)	5.5s	12.6s
s1423	74	36.6(37.3)	41	96.2(97.1)	41	**97.9(98.8)**	716s	1212s
s1488	6	97.0(100)	5	100(100)	**2**	99.9(100)	7.2s	44.6s
s1494	6	96.4(100)	5	99.2(100)	**2**	99.1(100)	7.3s	38.2s
s5378	179	70.2(73.8)	48	93.9(99.9)	48	**97.1**(99.9)	332s	415s
s9234	211	9.3(14.1)	97	90.5(96.2)	97	90.5(**96.3**)	11.6hrs	4.48hrs
s13207.1	638	9.2(88.5)	160	90.5(96.6)	160	**94.3(97.7)**	4.72hrs	4.8hrs
s15850.1	534	35.4(44.1)	296	**94.8(99.9)**	296	95.2(99.6)	27.2hrs	1.08hrs
s35932	1728	89.1(99.3)	306	89.8(100)	**232**	89.8(100)	12.9hrs	3565s
s38417*	1636	3.6(4.4)	516	84.7(85.5)	516	84.0(**90.2**)	8.2hrs	7.8hrs
s499	22	0(44.9)	22	100(100)	22	100(100)	255s	1.5s
s635	32	0(99.1)	31	99.9(100)	31	99.4(100)	73.3s	2.1s
s938	32	5.2(60.8)	30	74.0(99.4)	30	56.4(**99.5**)	82s	335s
s967	29	7.13(100)	5	100(100)	**4**	100(100)	4.7s	3.7s
s991	19	0(100)	7	3.2(100)	7	**99.9**(100)	24s	2.2s
s1269	37	17.9(58.7)	7	79.8(96.1)	7	**99.8(99.9)**	12s	82s
s1512	57	4.9(96.2)	29	**100(100)**	29	99.9(99.9)	236s	1466s
s3271	116	98.2(98.3)	26	**99.9(100)**	26	98.9(99.0)	49s	1941s
s3330	132	72.9(73.9)	7	**82.8(84.1)**	7	81.7(82.4)	39s	2961s
s3384	183	88.6(88.6)	19	88.6(88.6)	19	**89.5(89.5)**	229s	5.61hrs
s4863	104	96.2(96.4)	40	100(100)	40	99.9(100)	204s	117s
s6669	50	99.5(99.5)	17	99.6(99.6)	**7**	99.8(**99.9**)	40s	780s
am2910	87	89.2(96.3)	25	91.1(98.4)	**3**	**92.8(99.6)**	8.8s	1898s
div16	50	78.0(84.3)	18	**91.6(98.0)**	18	91.4(97.8)	280s	1.25hrs
mult16	55	87.2(88.0)	19	**99.77(100)**	19	99.4(99.8)	94.3s	175s
barcode	46	53.7(54.6)	27	99.8(100)	27	99.5(100)	638s	108s
gcd	49	82.6(82.7)	32	**98.6(98.7)**	24	94.9(95.0)	153s	1.78hrs