fied by inserting dummy bits where necessary. Case studies indicate that the logic overhead for the scan path can be reduced significantly using this partial scan methodology, particularly in pipelined circuits such as those which often occur in digital signal processing chips. By eliminating scan registers in the critical path, the performance of the circuit can also be enhanced.

References

[1] E. Trischler, "Design for testability using incomplete scan path and testability analysis," *Siemens Forsch.- u. Entwickl.-Ber.*, vol. 13, no. 2, pp. 56–61, 1984.

[2] H.-K. T. Ma, S. Devadas, A. R. Newton, and A. Sangiovanni-Vincentelli, "An incomplete scan design approach to test generation for sequential machines," in *Proc. IEEE Int. Test Conf.*, Sept. 1988, pp. 730–734.

[3] K.-T. Cheng and V. D. Agrawal, "An economical scan design for sequential logic test generation," in *Proc. Fault-Tolerant Comput. Symp. (FTCS-19)*, June 1989, pp. 28–35.

[4] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "A complete solution to the partial scan problem," in *Proc. IEEE Int. Test Conf.*, 1987, pp. 44–51.

[5] A. Kunzmann, "Produktionstest synchroner Schaltwerke auf der Basis von Pipeline-strukturen," in *Proc. 18. Jahrestagung der Gesellschaft für Informatik*, Hamburg, 1988, pp. 92–105, Informatik-Fachberichte 188, Springer-Verlag.

[6] A. Miczo, *Digital Logic Testing and Simulation*. New York: Harper and Row, 1986.

[7] R. Gupta, R.Gupta, and M. A. Breuer, "BALLAST: A methodology for partial scan design," in *Proc. Fault-Tolerant Comput. Symp. (FTCS-19)*, June 1989, pp. 118–125.

[8] —, "A methodology for partial scan design using balanced sequential structures," Tech. Rep. CRI-88-59, Univ. of Southern California, Dep. Elec. Eng.-Syst., Dec. 1988.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[10] H. W. Lenstra, "The acyclic subgraph problem," Tech. Rep. BW 26/73, Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam, July 1973.

[11] R. E. Tarjan, *Data Structures and Network Algorithms*. Philadelphia, PA: SIAM, 1983.

A Partial Scan Method for Sequential Circuits with Feedback

KWANG-TING CHENG AND VISHWANI D. AGRAWAL

Abstract—**This paper presents a method of partial scan design in which the selection of scan flip-flops is aimed at breaking up the cyclic structure of the circuit. Experimental data are given to show that the test generation complexity may grow exponentially with the length of cycles in the circuit. This complexity grows only linearly with the sequential depth. Graph-theoretic algorithms are presented to select a minimal set of flip-flops for eliminating cycles and for reducing the sequential depth. Tests for the resulting circuit are efficiently generated by a sequential logic test generator. An independent control of the scan clock allows insertion of scan sequences within the vector sequence produced by the test generator. We obtained a 98% fault coverage for a 5000 gate circuit by scanning just 5% of the flip-flops. A novel design of scan flip-flop reduces the input pin and signal routing overheads of the added scan clock.**

Index Terms—**Design for testability, scan design, sequential circuit testing, test generation.**

Manuscript received July 6, 1989; revised November 18, 1989. An earlier version of this paper appeared in the Proceedings of the 19th International Symposium on Fault-Tolerant Computing under the title, "An Economical Scan Design for Sequential Logic Test Generation."

The authors are with AT&T Bell Laboratories, Murray Hill, NJ 07974.

IEEE Log Number 8933873.

Introduction

Partial scan is an attractive method of designing VLSI circuits with guaranteed testability. Two approaches have been used. In the first approach [1], scan flip-flops are selected by testability analysis. A sequential logic test generator is then used to obtain test vectors. The scan overhead and fault coverage depend upon how well the testability analysis represents the problems of the test generator. The second approach [2] uses functional vectors and a combinational test generator to select scan flip-flops and to generate test vectors only for the faults not detected by the functional vectors. The scan overhead, in this case, may depend on the quality of functional vectors.

The motivation in the present work is to eliminate the reliance on testability analysis or functional vectors. We observed the performance of a sequential logic test generator that uses time-frame expansion of the circuit to generate tests by the *D*-algorithm. We found that the cycles in the circuit graph, where nodes are the flip-flops and the edges are the combinational signal dependencies, are mainly responsible for the test generation complexity. In the absence of such cycles, even large sequential depth causes no serious problem to the test generator.

Finite state machines that can be realized without a cyclic structure are known as *definite machines* [3]. These circuits behave like a pipeline and their test generation complexity is similar to that of a combinational circuit [4]. In general, however, a sequential circuit may have cycles for several reasons; its function may be nondefinite or cycle-free implementation may require too many extra gates and flip-flops. The main idea we present in this paper is that of breaking some or all cycles by scanning a minimal subset of flip-flops.

We use a graph-theoretic method to select a minimal set of flip-flops for scan. The clock of scan flip-flops is given a separate control to allow insertion of scan sequences between the vector sequences produced by the test generator. We also give an efficient flip-flop design for controlling the scan clock without any extra pin over what is needed for the normal scan design. The results show that for a large circuit, by scanning only 5% of the flip-flops, tests could be effectively generated automatically and without any interaction from the designer.

What Makes Test Generation Difficult?

The difficulty in generating tests for general sequential circuits arises due to the poor controllability and observability of memory elements. In practice, the level of this difficulty is circuit-dependent. Some circuits can be easily handled by a test generator while others take enormous computing effort. Table I shows the test generation results obtained from a sequential logic test generation program [5]. One circuit is a traffic light controller (TLC) and the other (CHIP-A) is a CMOS chip designed for a graphics terminal. Even though CHIP-A is three times larger than TLC, our test generator processed it in one-fifth the time and produced a significantly higher fault coverage. The CPU time in this table and for all other results given in this paper is for a VAX8650 computer.

The results in Table I show that the gate count is definitely not a key parameter in determining the complexity of test generation. Sequential depth which is often cited as a *measure of sequentiality* of a circuit does not explain the result either. If we define *sequential depth* as the largest number of flip-flops on a path between inputs and outputs, then it is 14 for both circuits. However, if we consider the *cycles* formed by flip-flops, CHIP-A contains no cycles greater than length one. A length one cycle (or loop) is formed when a flip-flop's output feeds back into its own input after passing only through combinational logic. TLC, on the other hand, has many longer cycles that may be responsible for the higher complexity.

Test Length and Cycles in Sequential Logic

Consider a directed graph $G = \{V, E\}$ of a synchronous sequential circuit. A vertex v_i in this graph represents a flip-flop (or state

TABLE I
TEST GENERATION FOR TWO SEQUENTIAL CIRCUITS

Circuit	No. of gates	No. of FFs	Test. gen. CPU sec.	Fault coverage
TLC	355	21	1246	89.01%
CHIP-A	1112	39	269	98.80%

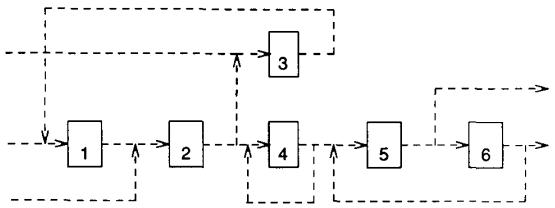

Fig. 1. A circuit with six flip-flops.

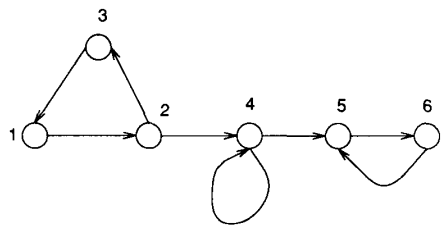

Fig. 2. Graph of the circuit in Fig. 1.

variable) i. A directed edge from vertex v_i to vertex v_j implies a combinational path from flip-flop i to flop-flop j. Only the memory elements and the dependencies between memory elements are explicitly represented in this graph. Fig. 1 illustrates a circuit with six flip-flops. Signal flow through combinational elements is represented by dashed lines. The circuit is synchronous and all flip-flops are controlled by a single clock signal which is not shown. The graph of this circuit is shown in Fig. 2.

We can classify the directed graph of a circuit into three categories: 1) acyclic directed graph, 2) directed graph with only self-loops, and 3) directed graph with cycles of two or more vertices. The effect of cycles on testing complexity was recognized by early workers [3]. Our analysis may be an oversimplification but still provides somewhat of an average complexity of test generation. A more rigorous testability analysis of cycles may be found in a recent paper [6].

Let us define the *distance* along a directed path between two vertices as the number of vertices on that path. The *sequential depth* of the circuit is the distance along the longest path in its graph. For example, the sequential depth of the circuit in Fig. 1 is five. If the graph is acyclic, i.e., there is no feedback among sequential elements, then we can levelize the graph. It can be shown that for a circuit with depth D, any single fault can be tested by at most D test vectors [4]. Furthermore, the size of a complete test set is bounded by $D \cdot 2^n$ where n is the number of primary inputs. Obviously, this type of sequential circuit will be easy to analyze by a sequential test generator.

Consider the second type of structure, i.e., circuits with only self-loops. In this case, the output of a flip-flop may link to its input through combinational logic. For these flip-flops, we can gather up all combinational logic between the output and input of the flip-flop to create a two-state finite state machine (FSM) and use a single vertex to represent this FSM by a reconstructed graph. Thus, in the new representation, a vertex can either represent a generalized two-state machine or a single flip-flop depending on whether or not the corresponding flip-flop has a self-loop. The new directed graph is acyclic again. However, justifying a logic value at the output of a vertex, which represents a generalized two-state machine, may need two patterns at the inputs of the vertex. In contrast, only one pattern is needed for a vertex which represents a single flip-flop without self-loop. This follows from the assumption that the two-state machine is initializable to some given state by one pattern. If this initial state is different from the required state, then one more pattern will be needed to move the machine to the other state. Using the pipeline argument as before, a test can be constructed with at most $2 \cdot D$ vectors.

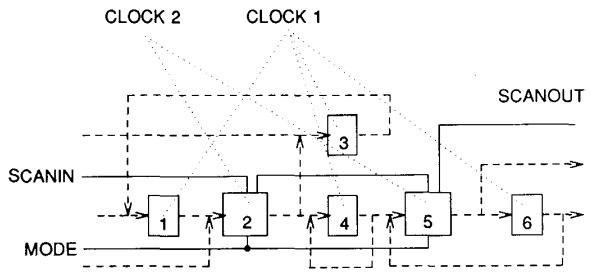

Fig. 3. Design with flip-flops 2 and 5 scanned.

For the third type of graph, with longer cycles, we again reconstruct an acyclic graph. A vertex in this new graph is a finite-state machine that can have up to 2^L states where L is the maximum length of any cycle. We assume that a vertex can be initialized to some state by a vector. To set the node in the desired state then may take at most $2^L - 1$ vectors. In general, the depth of the reconstructed graph will be less than D. However, D is still an upper bound for depth. Thus, the length of a test sequence can be $D \cdot 2^L$. Most faults in the circuit may be tested by shorter sequences. We must remember that our derivation assumes a specific initialization procedure. Also, different test generators may pack vectors differently while creating test sequences. This sequence length formula should, therefore, be considered only an average measure of the test length. Our experience shows that the actual length can be larger in a few cases. A theoretical upper bound would be much higher.

The above analysis shows that the test length is related to the cycle length defined as the maximum number of flip-flops in a feedback cycle. If a sequential circuit test generator employing the time frame expansion technique is used, the number of processed time frames is equal to the test length. The longer the test length, the longer is the test generation time, and the higher is the probability that the fault will be dropped due to some practical time limit imposed on the program. To support this argument, we analyzed the structure of the two circuits in Table I. CHIP-A's cycle length is 1 and that of TLC is 4. It is also found that the longest test for a single fault in TLC has 243 vectors. The sequential depth for this circuit is 14 and it has cycles of length 4. Our formula, thus, gives a test length of 224. The longest sequence for CHIP-A has 102 vectors. Obviously, TLC presents a more difficult test generation problem than CHIP-A. The high fault coverage and low computation time for CHIP-A also indicate that self-loop structure can be handled successfully by a sequential test generator.

A DESIGN FOR TESTABILITY METHOD

Based on the above observations, we present a partial scan methodology.

Scan Register Design: A scan flip-flop is assumed to break the circuit for test generation by making its I/O signals observable and controllable like primary outputs and inputs. This assumption is justified for a partial scan circuit if a separate control on the clock of scan flip-flops is available. To reduce the test generation complexity, we select a minimal set of flip-flops to eliminate all cycles. Self-loops or cycles of unit length are not broken to keep the scan overhead low since the number of self-loops in some circuits can be quite large.

A simple example of this design is shown in Fig. 3. The original circuit (Fig. 1) has only one clock which is shown as CLOCK1 here. It is clear from Fig. 2 that if flip-flops 2 and 5 are scanned then the circuit will not have any cycles other than the self-loop of flip-flop 4. In Fig. 3, flip-flops 2 and 5 have been replaced by scan flip-flops (shown as larger blocks) and SCANIN, SCANOUT,

TABLE II
RESULTS OF *BreakLoop*

Circuit	Total No. of FFs	No. of scan FFs to break all cycles*	CPU time
MULT4	15	5	5 sec.
TLC	21	9	36 sec.
CHIP-B	318	14	5.5 min.†

* Except self-loops.
† Three-pass run, 2 min. limit per pass.

and MODE signals have been incorporated [7]. A new clock signal, CLOCK2, controls the scan flip-flops. In the normal mode of the circuit, CLOCK1 and CLOCK2 will be identical signals. However, in the scan mode, CLOCK2 alone will be activated. Thus, while the states of scan flip-flops are being shifted in, all other flip-flops will retain their states.

Flip-Flop Selection Algorithm: A directed graph is first constructed from the circuit netlist. All self-loops are removed. The remaining graph, thus, contains either no cycle or cycles longer than 1. A program called *BreakLoop* is used to select a small subset of vertices to break all cycles. Since the problem of finding the vertex set that will break all cycles (feedback vertex set problem) is NP-complete [8], heuristics must be used to bound the computation time. In the program *BreakLoop*, the process of finding cycles and selecting vertices is divided into several passes. In each pass, a time limit is set for the cycle finding process. When the time limit is reached, the search for more cycles is suspended for the current pass. The selection of flip-flops at this point is based on the list of cycles found in the pass. Notice that breaking a cycle will automatically break all bigger cycles in which this cycle is embedded. Therefore, the cycle list should only contain a set of *representative* cycles such that no cycle is embedded in any other cycle. In order to generate such a cycle list, when a new cycle is found, the current cycle list is first searched to check whether any cycle in the list covers or is covered by the new cycle. If there exists any cycle in the list embedded in the new cycle, the new one is simply discarded. If the new cycle is embedded in a cycle that is already in the list, then the new one replaces the existing cycle. If neither case is found, the new cycle is added to the list. The algorithm for selecting vertices to break all cycles is as follows:

```
repeat
   for every vertex
      count the frequency of appearance
         in the cycle list.
   select the most frequently used vertex.
   remove all cycles containing the selected
      vertex from the cycle list.
until (cycle list is empty).
```

The selected vertices form a minimal set of vertices that break all cycles in the cycle list. These vertices are removed from the directed graph and the graph is reconstructed. A new pass is then run. This process continues until the reconstructed graph becomes acyclic.

The performance of this heuristic is shown in Table II. MULT4 is a 4-bit multiplier circuit with 15 flip-flops and 382 gates. All cycles (except self-loops) can be broken by scanning 5 flip-flops. All cycles in TLC can be broken by scanning 9 out of 21 flip-flops. CHIP-B is a VLSI chip with 5294 gates and 318 flip-flops. When the time limit for a single pass was set to 2 min, all cycles were broken in three passes with 14 flip-flops selected for scan.

Another program, *BreakPath*, selects flip-flops to reduce the sequential depth of the circuit. In *BreakPath*, all paths longer than a user-specified limit, *PathThreshold*, are first recorded. For each flip-flop, the frequency of usage in the path list is then calculated. For a path of length D, if we scan the ith flip-flop, then the path will be broken into two shorter paths of length $i-1$ and $D-i$, respectively. Clearly, selecting the flip-flop at the center of a path is the best choice for reducing the sequential depth of the circuit. Therefore, in calculating the usage frequency, the flip-flops closer to the center of a path are weighted higher. The most frequently used flip-flop is selected and the path list is updated. This procedure repeats until the lengths of all paths in the list are smaller than *PathThreshold*.

TABLE III
PARTIAL SCAN FOR TLC (335 GATES, 21 FLIP-FLOPS)

No. of scan FFs	Max. cycle length	Depth	CPU sec. Test gen.	CPU sec. Fault sim.	Fault cov.	No. of tests	Total Vectors
0	4	14	1246	61	89.01%	805	805
4	2	10	157	11	95.90%	247	988
9	1	5	32	4	99.20%	136	1224
10	1	3	13	4	100.00%	112	1120
21	0	0	2	2	100.00%	52	1092

Test Generation: In order to effectively use the partial scan register for generating and applying tests, a separate scan clock is used. While shifting bits in or out of the scan register, the normal clock is held inactive to hold the states of the nonscanned flip-flops unchanged. An economical design of a scan flip-flop that combines the *scan clock* input with *mode* input is described later. With this design, the scan clock operation would not require any extra input pin over what is needed for the normal scan design.

After the scan flip-flops have been selected, the netlist is modified to create a test generation model. In this model, all scan flip-flops are removed and their input and output signals are added to the primary output and primary input lists, respectively. A sequential circuit test generator and a fault simulator are used to generate tests for faults in the modified netlist model. The test generator picks a fault from the fault list and produces a vector sequence which is run through the fault simulator to eliminate all other detected faults from the fault list. The test generator next picks another fault from the updated fault list and the process is repeated until either the fault list becomes empty or is reduced to a suitably small size.

The vector sequences, thus generated, are converted into scan sequences by adding two types of vectors. First, scan register tests are added. These tests are run before applying any vector sequences produced by the test generator. They run entirely in the scan mode and are designed to cover all faults in the scan register. Once the correct operation of the scan register is ensured, other scan tests are applied. These scan tests are generated from the vector sequences produced by the test generator. Each vector is preceded by a scan-in sequence to set the states of scanned flip-flops. A scan-out sequence is added at the end of each vector sequence.

EXPERIMENTAL RESULTS

The TLC circuit has a sequential depth 14 and it contains cycles of length upto 4. Algorithm *BreakLoop* selected 9 flip-flops to break all cycles greater than length 1. When these flip-flops are scanned the sequential depth of the circuit becomes 5. This can be reduced to 3 by scanning 10 flip-flops. Of course, if all 21 flip-flops are scanned, the circuit is like a combinational circuit for which the depth is 0. The results of test generation are shown in Table III. The reduction in test generation complexity with the increase in the number of scan flip-flops is evident from the "test gen. CPU Sec." and "fault coverage" shown in the table. The reported CPU time was recorded on a VAX-8650. Considering the fault coverage and scan overhead as the most important design criteria, the best result is obtained by simply eliminating all cycles (excluding self-loops). An interesting illustration of the effectiveness of this method of flip-flop selection is given in Fig. 4. The test length here is the length of a vector sequence generated for a target fault. The number of faults corresponds to the faults found detectable by the vector sequence through fault simulation.

Table IV gives the results for the MULT4 circuit discussed earlier. Eliminating cycles increased the fault coverage by 1.67%. Table V shows the effect of reducing the sequential depth in CHIP-A. The circuit does not have any cycles greater than length 1 and while the circuit is larger than TLC and MULT4, a reasonably high fault coverage is obtained even without scan. The test length statistics for the CHIP-A circuit showed that the maximum test length was somewhat proportional to the sequential depth.

Our last example is the large circuit, CHIP-B. Its sequential depth is 43 and it also has cycles. *BreakLoop* identified 14 out of 318 flip-flops. This reduced the sequential depth to 19. As shown in Table VI, the improvement is significant; fault coverage is increased by

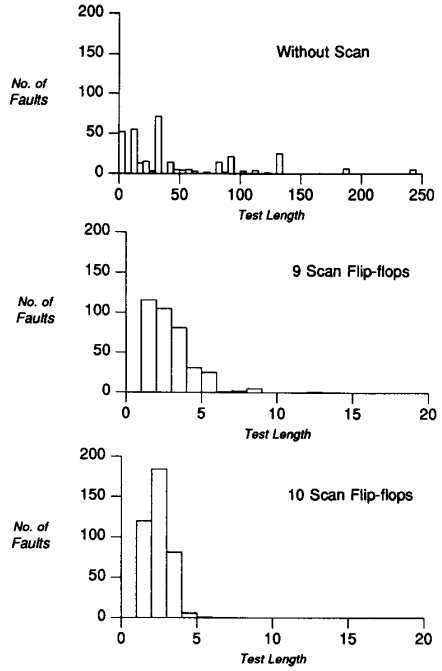

Fig. 4. Test length statistics for the TLC circuit.

TABLE IV
PARTIAL SCAN FOR MULT4 (382 GATES, 15 FLIP-FLOPS)

No. of scan FFs	Max. cycle length	Depth	CPU sec. Test gen.	CPU sec. Fault sim.	Fault cov.	No. of tests	Total Vectors
0	4	13	75	5	98.01%	115	115
5	1	6	8	2	99.68%	69	345
6	1	4	8	2	99.68%	72	432

TABLE V
PARTIAL SCAN FOR CHIP-A (1112 GATES, 39 FLIP-FLOPS)

No. of scan FFs	Max. cycle length	Depth	CPU sec. Test gen.	CPU sec. Fault sim.	Fault cov.	No. of tests	Total Vectors
0	1	14	269	274	98.80%	868	868
8	1	10	85	56	99.60%	529	4232
16	1	6	49	33	99.80%	387	6192

15% and test generation time is reduced by a factor of 3. Adding more scan flip-flops to reduce the sequential depth somewhat linearly reduces the test generation time. However, further increase in fault coverage is only marginal. To reduce the memory requirement of the fault simulator, most of the results in Table VI were obtained by processing a 20% sample of faults. The last case of 87 scan flip-flops was repeated with all faults.

Test length statistics for this circuit showed that for the unscanned design, the test length was smaller than the sequential depth of 43. This is because the test generator, due to its time limit, was not able to produce tests for faults requiring very long sequences as evidenced by the 82.6% fault coverage. Eliminating cycles made the test generation manageable at a very small cost of scanning just 5% of flip-flops.

We further studied the cyclic structure of some available benchmark circuits [9]. The results were reported in [10]. We selected four circuits that have complex cyclic structure for our partial scan experiment. The test generation results are listed in Table VII. The circuit name (first column) conveys the size of the circuit. These circuits require between 18% to 43% scan flip-flops to achieve fault coverages higher than 99%. Interestingly, larger circuits need a smaller fraction of flip-flops to be scanned.

TABLE VI
PARTIAL SCAN FOR CHIP-B (5294 GATES, 318 FLIP-FLOPS)

No. of scan FFs	Max. cycle length	Depth	CPU sec. Test gen.	CPU sec. Fault sim.	Fault cov.	No. of tests	Total Vectors
0	40	43	11018*	2256	82.60%	948	948
14	1	19	2946*	2986	97.90%	2607	36498
36	1	10	2041*	2765	98.30%	2494	89784
44	1	6	1207*	2526	97.80%	1741	76604
87	1	4	643*	862	98.20%	842	73254
87	1	4	2294	7961	98.43%	2536	220632

* 20% sample of total faults used for test gen. and fault sim.

FLIP-FLOPS FOR PARTIAL SCAN DESIGN

Two types of flip-flops have been used in scan design. In one type, known as the level-sensitive design [11], two nonoverlapping clock signals are used in the normal mode operation of the master and slave latches. A third clock signal, called the scan clock, and the normal mode slave clock are used together to accomplish the scan operation. In a partial scan design, only a selected subset of flip-flops will receive the scan clock signal. The scan operation, therefore, will not disturb the state of unscanned flip-flops. Compared to a complete

TABLE VII
TEST GENERATION FOR SEQUENTIAL BENCHMARK CIRCUITS WITH PARTIAL SCAN

Circuit Name	Total No. of FFs	Scan FFs No.	Scan FFs %	No. of Test Vectors	Fault Coverage (%) Tested	Fault Coverage (%) Redundant	Fault Coverage (%) Total	Tgen + Fsim Sec. (VAX 8650)
s400	21	9	42.86	107	98.11	1.89	100.00	7
s713	19	7	36.84	83	90.71	9.29	100.00	18
s5378	179	32	17.89	2612	93.38	6.32	99.70	1253
s9234	228	53	23.25	3458	43.23	55.80	99.04	6208

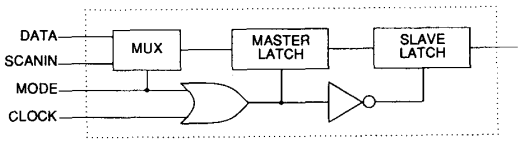

Fig. 5. Scan flip-flop for single clock design.

scan design, the overhead in partial scan design is thus reduced in direct proportion to the fraction of flip-flops that are scanned.

An alternative design practice, however, uses a single clock signal. Either a master–slave operation is achieved by using two latches that are sensitive to different levels of the clock signal, or simply edge-triggered latches are used. For scan design, flip-flops are preceded by multiplexers to switch between the normal signal and the scan signal [7]. All multiplexers are controlled by a mode control signal that is added to the circuit. In this design, the same clock signal is used for both normal and scan functions. In a partial scan circuit, the unscanned latches must hold their normal states while data are being scanned in. One way to accomplish this is to use an additional clock pin feeding only the scanned flip-flops. In the normal mode, both clock pins will carry the same signal. In the scan mode, the clock signal is applied only to the second clock pin. Thus, in addition to the other signals needed for a complete scan design, the partial scan design may need one extra clock pin. The extra clock pin can be eliminated if the scan flip-flop, shown in Fig. 5, is used. The two signals, MODE and CLOCK, are routed to all scanned flip-flops. The same CLOCK signal is routed to all unscanned flip-flops.

Signal waveforms for the two modes are easily derived. Assume that the latches in Fig. 5 are active when their clock input is *high*. Also, the multiplexer selects DATA when its control signal is *low*, otherwise it selects SCANIN. In the normal mode, MODE is held *low* and CLOCK carries a periodic *high* (master active) and *low* (slave active) signal. Thus, DATA inputs are clocked in all flip-flops. In the scan mode, CLOCK is held *low*. Thus, all unscanned flip-flops, that do not receive MODE, are unaffected. For scanned flip-flops, MODE now carries a clock-like (periodic high and low) signal which activates their master and slave latches. Also, when the master latch is active, that is, MODE is *high*, SCANIN is selected by the multiplexer. In this design, the need for any additional clock pin is eliminated [12].

CONCLUSION

We have shown that the cyclic structure of a synchronous sequential circuit is mainly responsible for the difficulty of test generation. Partial scan design that breaks up the cycles significantly improves the performance of a sequential circuit test generator. We were able to generate all tests completely automatically for a complex sequential circuit by scanning just 5% of all flip-flops. Beside achieving a low overhead, this partial scan concept is more attractive for two other reasons: 1) the selection of scan flip-flops does not rely on testability measures but directly reduces the circuit complexity for the test generator, and 2) the reliance on functional vectors, that may not always be available, is eliminated.

REFERENCES

[1] E. Trischler, "Incomplete scan path with an automatic test generation methodology," in *Proc. Int. Test Conf.*, Nov. 1980, pp. 153–162.
[2] V. D. Agrawal, K. T. Cheng, D. D. Johnson, and T. Lin, "Designing circuits with partial scan," *IEEE Design Test Comput.*, vol. 5, pp. 8–15, Apr. 1988.
[3] A. D. Friedman and P. R. Menon, *Theory and Design of Switching Circuits*. Rockville, MD: Computer Science Press, 1975.
[4] A. Miczo, *Digital Logic Testing and Simulation*. New York: Harper and Row, 1986.
[5] W. T. Cheng, "The BACK algorithm for sequential test generation," in *Proc. Int. Conf. Comput. Design (ICCD-88)*, Rye Brook, NY, Oct. 1988, pp. 66–69.
[6] R. V. Hudli and S. C. Seth, "Testability analysis of synchronous sequential circuits based on structure data," in *Proc. Int. Test Conf.*, Aug. 1989, pp. 364–372.
[7] V. D. Agrawal, S. K. Jain, and D. M. Singer, "Automation in design for testability," in *Proc. Custom Integrated Circuits Conf.*, Rochester, NY, May 1984, pp. 159–163.
[8] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974, ch. 10.
[9] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profile of sequential benchmark circuits," in *Proc. Int. Symp. Circuits Syst.*, May 1989, pp. 1929–1934.
[10] K. T. Cheng and V. D. Agrawal, "Concurrent test generation and design for testability," in *Proc. Int. Symp. Circuits Syst.*, May 1989, pp. 1935–1938.
[11] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," *J. Des. Automat. Fault Tolerant Comput.*, vol. 2, pp. 165–178, May 1978.
[12] M. R. Mercer and V. D. Agrawal, "A novel clocking technique for VLSI circuit testability," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 207–212, Apr. 1984.

High-Performance Fault-Tolerant VLSI Systems Using Micro Rollback

YUVAL TAMIR AND MARC TREMBLAY

Abstract—**A key to achieving a high degree of fault tolerance is the ability to detect errors as soon as they occur and prevent erroneous information from spreading throughout the system. In highly reliable systems, this is usually accomplished by checkers and isolation circuits in the communication paths from each module to the rest of the system. This additional circuitry reduces performance by requiring either longer clock cycles or additional pipeline stages. We present a technique, called *micro rollback*, which allows most of the performance penalty for concurrent error detection to be eliminated. Detection is performed *in parallel* with the transmission of information between modules, thus**

Manuscript received July 13, 1989; revised November 20, 1989. This work was sponsored by the SDIO Innovative Science and Technology Office, managed by the Office of Naval Research, contracted to the Jet Propulsion Laboratory under task plan 80-2984; and by Hughes Aircraft Company and the State of California MICRO program. M. Tremblay is supported by an IBM fellowship.

The authors are with the Department of Computer Science, University of California, Los Angeles, CA 90024.

IEEE Log Number 8933874.