Improving Circuit Testability by Clock Control

Kent L. Einspahr
Concordia College, eins@seward.ccsn.edu

Sharad C. Seth
University of Nebraska-Lincoln, seth@cse.unl.edu

Vishwani D. Agrawal
AT&T Bell Laboratories, Murray Hill, NJ

Proceedings., Sixth Great Lakes Symposium on VLSI, 1996. doi: 10.1109/GLSV.1996.497635

Improving Circuit Testability by Clock Control

Kent L. Einspahr
Dept. of Computer Science
Concordia College
Seward, NE 68434-1599
eins@seward.ccsn.edu

Sharad C. Seth
Dept. of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
seth@cse.unl.edu

Vishwani D. Agrawal
AT&T Bell Laboratories
Murray Hill, NJ 07974-0636
va@research.att.com

Abstract

The testability of a sequential circuit can be improved by controlling the clocks of individual storage elements during testing. We propose several clock control strategies derived from an analysis of the circuit, its S-graph structure, and its function. Through examples we show how the number of clocks affects the circuit's testability. It is shown that if certain flip-flops (FFs) are scanned (or otherwise initialized), the remaining FFs can be controlled and initialized to any arbitrary state using the clock control. We derive a controllability graph *and use it to assign clocks to FFs and to schedule the clocks to set the FFs to an arbitrary state during test. Our analysis of sequential benchmark circuits indicates that this could be an attractive scheme for combining partial scan with clock control.*

1 Introduction

A number of recent papers have considered the problem of applying clock control to groups of flip-flops in a sequential circuit to improve the testability of the circuit. The basic idea of this approach to testability involves dividing the flip-flops of the circuit into groups, with each group controlled by its own clock. During the test mode one or more clocks may be enabled activating only those flip-flops controlled by the clock(s). The *normal* clock mode in which all partitions are active may be realized by simultaneously enabling all clocks. The theoretical foundations for this multiple clock control were initially presented in [1] and [6]. Different variations have been suggested in [2], [7], and [9]. A dynamic approach to clock control has been proposed by Baeg & Rogers [2]. A clock control method for delay testing is presented by Fang & Gupta [7]. Rajan, *et al.* [9] control the enable lines of the FFs to provide clock control by identifying states that are difficult to reach but are necessary to detect some faults.

While the previous research has presented results indicating the merits of clock control, little work has been reported on the improved testability in relation to the number of clocks used and the specific partitioning of the flip-flops. The purpose of this paper is to investigate those issues.

We show that if certain FFs are *controllable* (i.e., can be set to either 1 or 0 independent of the states of other FFs) then the circuit can be initialized to an arbitrary state, thus providing controllability equivalent to full-scan. If the circuit is not so controllable then it is made controllable by a combined partial-scan/clock-control scheme for testability improvement. We present an algorithm that finds a satisfactory clock assignment for FFs so that the circuit can be controlled (under clock control). In contrast to [9], our method considers the functions of FF inputs to determine partitions and the sequence of steps to bring each FF to a specified state. This DFT scheme may be an attractive alternative to existing DFT schemes for sequential circuits.

2 Testability using clock control

We assume that there are n FFs in the circuit which are partitioned into m groups, where $m \leq n$. Each group of FFs is assumed to have its own independent clock control. During any clock cycle, it is possible to enable any number of clock signals. For simplicity,

0-8186-7502-0/96 $5.00 © 1996 IEEE

in the present work we activate a single group of FFs at any one time and disable all others. Our objective is to use the clocks to control the state of each FF *independent* of the state of other FFs.

2.1 Hardware implementation for clock control

The clock control model can be implemented in hardware by using a clock distribution tree in which a demultiplexer splits the clock line into m clocks as shown in the schematic in Figure 1. A single test mode line determines the circuit mode (0 = normal mode, 1 = test mode) and $\lceil \log m \rceil$ select lines implement the clock control. In the normal mode all clock lines are enabled. In the test mode a single clock line (determined by the select lines) is enabled.

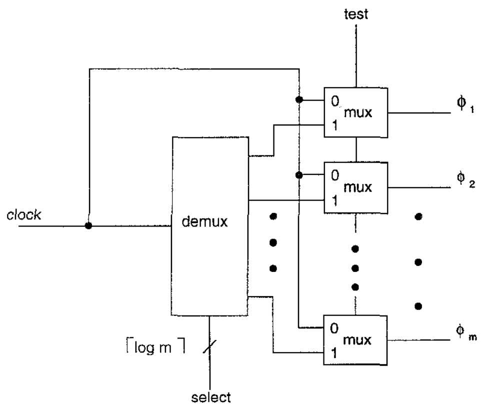


Figure 1. Clock control implementation.

2.2 Number of clocks vs. testability

The testability of the circuit improves with the number of independent clocks. This is readily demonstrated for a mod-2^n binary counter such as the mod-8 binary counter shown in Figure 2. The normal mode is shown in state diagram A. Figure 2B shows one possible grouping of bits for a two-clock control. The dashed edges are the additional transitions that become possible when one or the other clock is disabled. With the additional transitions, it becomes easier to navigate from one state to another. For example, the average and maximum path lengths for the normal mode in Figure 2A are 4.0 and 7, respectively. The same measures for the two-clock control in Figure 2B are 2.3 and 4. Similarly, we find that the average path length under 3-clock control, in which each bit is controlled by a separate clock, improves to 2.1.

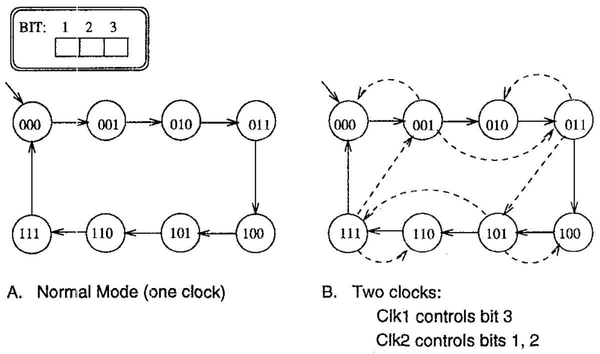


Figure 2. Clock control in binary counter.

In general, for the mod-2^n counter, the normal mode average and maximum path lengths are easily derived to be 2^{n-1} and $2^n - 1$, respectively. In other cases, the path-lengths can be estimated by noting that with m clocks, it is possible to ***break*** the counter into m subcounters of no more than $\lceil n \div m \rceil$ stages. Thus, the maximum path length for each subcounter will be $O(2^{n/m})$ and the maximum path length of the whole counter will be $O(m \cdot 2^{n/m})$.

We will show that in the extreme case of n clocks the binary counter can be brought to any state in $O(n)$ time starting from any arbitrary state. The only assumption we make is that the FF corresponding to the least significant bit (LSB) is initially controllable.

Suppose the starting state is $(s_1, s_2, \cdots, s_n)$ and the desired state is $F = (f_1, f_2, \cdots, f_n)$. Note that a binary counter stage toggles whenever all lower order bits are 1. Hence, a two-pass process to reach the desired state from an arbitrary initial state is possible. We assume that the stages are numbered $1, \cdots, n$ from LSB to MSB (most significant bit) and that stage i is controlled by clock ϕ_i. In the first pass, the counter is initialized to the all-1 state starting from the LSB which is controllable by our assumption. Beginning at the unknown state $(y_1, \cdots, y_n) = (X, X, \cdots, X)$, successive steps during the first pass yield the states $(1, X, \cdots, X)$, $(1, 1, \cdots, X)$, $\cdots$, $(1, 1, \cdots, 1)$.

In the second pass, the bits in the final state F that differ from the all-1 state are changed sequentially, starting from the MSB. Successive steps yield the states $(1, 1, \cdots, 1, f_n)$, $(1, 1, \cdots, f_{n-1}, f_n)$, $\cdots$, $(f_1, f_2 \cdots, f_{n-1}, f_n)$. The initializing sequence for the counter requires $O(2n) = O(n)$ steps. Thus, in essence, the n-clock case is comparable to full-scan for this example.

In generalizing the above scheme, we notice that any standard implementation of the binary counter has the S-graph shown in Figure 3. The S-graph shows connectivity of FFs through combinational paths [4]. It has a node for each FF and there is a directed edge from the FF_i node to the FF_j node if FF_j can be reached from FF_i through a path involving only combinational logic elements. An extension of the S-graph can be defined which includes primary inputs as additional nodes. The graph reveals that each stage depends on itself and all preceding stages. Therefore, if the LSB stage is controlled and each subsequent stage can be independently controlled, the entire circuit can eventually be controlled by a process similar to above.

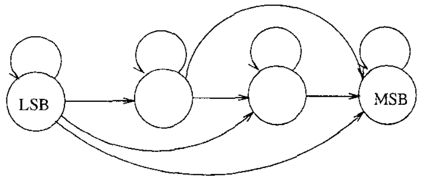


Figure 3. The S-graph of a binary counter.

3 Controllability graph (CG)

We limit the discussion to D-type flip-flops. A FF may be controlled in several ways: 1) from the primary inputs (PIs), 2) by an independent asynchronous set or reset, 3) by controlling the FF from previously controlled FFs [5], or 4) by scanning the FF. A *circuit is controllable* if each of its FFs can be controlled by one of the four methods.

We define an edge-labeled directed *generalized controllability graph* (GCG) that captures the controllability relation: $GCG = (V, E, W)$ where $V = \{$ all PIs, all FFs $\}$. The edge $(i, j) \in E$ and the condition (edge label) $c_{ij} \in W$ represent the functional dependence of node j on node i.

The condition c_{ij} is determined by analyzing the function of the input signal of FF_j, $f_j(x_1, \cdots, x_n)$ where the x_i's include the PIs and the pseudo inputs that occur in the formula for FF_j. Using the Shannon expansion,

$$\begin{aligned} f_j = \; & x_i \cdot f_j(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n) \; + \\ & \overline{x_i} \cdot f_j(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n) \end{aligned}$$

we define $f_j^{i=y} = f_j(x_1, \cdots, x_{i-1}, y, x_{i+1}, \cdots, x_n)$ where $y = 0, 1$. The state of FF_j is determined by

$$f_j = \begin{cases} x_i & \text{when } f_j^{i=1} \cdot \overline{f_j^{i=0}} \neq 0 \\ \overline{x_i} & \text{when } \overline{f_j^{i=1}} \cdot f_j^{i=0} \neq 0 \\ \text{independent of } x & \text{otherwise} \end{cases}$$

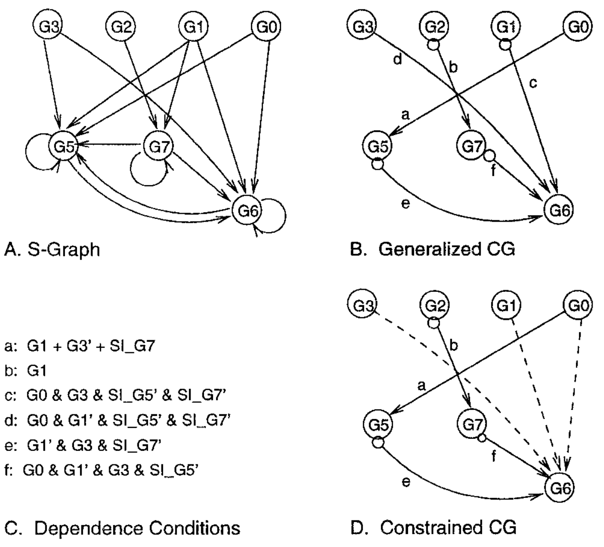
If either of the first two cases in the equation for f_j are satisfiable then FF_j can be controlled from x_i under the conditions $c^1_{i,j} = f_j^{i=1} \cdot \overline{f_j^{i=0}}$ and $c^0_{i,j} = \overline{f_j^{i=1}} \cdot f_j^{i=0}$, respectively, and a corresponding (directed) edge is added to the GCG. If neither of the first two cases is satisfiable, FF_j is not controllable from x_i.

We further restrict the conditions of the first two cases to be satisfiable independent of the previous state of FF_j since we may want to initially control FF_j from an unknown state. If the condition is not independent of the pseudo input of FF_j we may not be able to satisfy the condition without first controlling FF_j. Therefore, $E = \{ (i, j) \mid i, j \in V$ such that i can control j independent of the previous state of $j\}$, and $W = \{ c_{ij} \mid i, j \in V$ where c_{ij} is a condition under which i controls $j\}$.

An edge between two nodes exists in the GCG only if there is an edge between the same nodes in the S-graph (extended to include PIs) as shown for the ISCAS-89 benchmark circuit s27 in Figures 4A and 4B (the PIs are nodes G0 through G3). Note that there may be two edges between nodes in the GCG if both $c^1_{i,j} \neq 0$ and $c^0_{i,j} \neq 0$. As shown on edges b, c, e, and f in Figure 4B, an inversion bubble is used to differentiate $c^0_{i,j}$ edges from $c^1_{i,j}$ edges. The dependence condition for each edge in the GCG has been given as a reference to the table in Figure 4C. The dependence conditions are given in the sum of products (SOP) form in which the current state of FF Gi is referred to as a pseudo input and denoted SI_Gi for $i = 5, 6, 7$.

Next we place *constraints* on the edges in the GCG such that FF_i is controlled before it can control FF_j and all variables occurring in at least one product term of c_{ij} are also controlled. Hereafter, we will refer to this *constrained CG* simply as the CG.

The CG for s27 in Figure 4D is derived as follows. Initially, only PIs G0 through G3 are controllable. At the second level we include only those edges which represent controllability from nodes in the first level, i.e., edges a and b to FFs G5 and G7. We can eliminate edges c and d because they include the uncontrolled pseudo inputs SI_G5 and SI_G7. At the third level, since G5 and G7 can now be controlled, we can include edges e and f to G6. As a result, edges c and d are considered implicitly. We have included dashed edges in the CG to indicate the dependence of G6 on nodes G0, G1, and G3.



a: G1 + G3' + SI_G7
b: G1
c: G0 & G3 & SI_G5' & SI_G7'
d: G0 & G1' & SI_G5' & SI_G7'
e: G1' & G3 & SI_G7'
f: G0 & G1' & G3 & SI_G5'

Figure 4. s27 graphs.

4 Controllability algorithm

We propose a general procedure in Figure 5 for initializing the circuit to an arbitrary state using a combination of clock control and partial scan. The algorithm has three parts: generation of the controllability graph, clock assignment for each of the FFs, and clock scheduling to bring the circuit to the desired state.

To generate the controllability graph, we place all FFs and PIs into the initial CG and calculate the dependence conditions for all edges in the S-graph. As described above, we iteratively determine whether a FF can be controlled from PIs or other FFs using the constraints of the CG. For each FF that can be controlled, we add the respective edges to the CG and mark the FF "controlled". An edge is not included in the CG if its associated dependence condition requires controllability of a node at a higher level. If we are unable to extend the edge set for the CG and have uncontrolled FFs remaining, we heuristically select one (or more) of the uncontrolled FFs to be scanned. In the present context, this leads to combining clock control with partial scan in a hybrid DFT scheme.

4.1 Clock assignment

The objective in clock assignment is to partition the FFs into a minimum number of groups. To this end, we introduce two additional types of constraints:

```
Generate CG
  Initialize: $CG = \{FFs, PIs\}$; Calculate dependencies

  While circuit is not controllable {
    For each uncontrolled $FF_j$ {
      For each edge $(i,j)$ in S-graph where $i$ is controlled {
        If $FF_j$ is controllable from $i$ {
          Insert (i,j); Mark $FF_j$ controlled }
    } } }

    If no new edges were added to CG then
       Select scan FF(s) & Mark as controllable
  }

Clock Assignment
  Initialize: Clock groups $C_0$ = { Scan FFs }, $C_1 = \emptyset$
              Clock counter $m = 1$; Frontier $F, F' = \emptyset$
              Neighbor set $N$ = { neighbors of PIs & Scan FFs }
  While all FFs not controlled {
    While clock group $C_m$ not done {
      Move FFs in N satisfying constraints to $F$; $F' = \emptyset$
      If $F = \emptyset$ then clock group $C_m$ is done
      Else {
        Find max. length chains satisfying constraints of $C_m$
          (Chains have the form $FF_c \rightarrow FF_f \rightarrow \cdots$)
          ($FF_c$ is controllable; $FF_f \in F$)
        Move selected FF chain to $C_m$
        Move FFs in F no longer satisfying $C_m$ to F'
        $N = N \cup$ { unmarked neighbors of $FF_a$ }; Mark nodes
    } }
    Initialize next clock group: $m = m + 1, C_m = \emptyset$
    Restore FFs in F' to F
  }
  $maxclock = m - 1$

Clock Scheduling
  Set $m = maxclock$;
  While $m \neq 0$ {
    Set FFs in $C_m$ to (final) desired state; $m' = m$
    While $m' \neq 0$ {
      Bring parent FFs to necessary state
      $m' = m' - 1$
    }
    $m = m - 1$
  }
```

Figure 5. Controllability algorithm.

Chain assignment: *If FF_i controls FF_j under the condition c_{ij} and FF_j controls FF_k under the condition c_{jk} then FF_j and FF_k can be assigned to the same chain only if c_{ij} and c_{jk} can be satisfied simultaneously, i.e., $c_{ij} \cdot c_{kl} \neq 0$.*

Group assignment: *If FF_i controls FF_k and FF_j controls FF_ℓ (where $i = j$ is allowed) then FF_k and FF_ℓ can be assigned to the same clock group only if neither condition c_{ik} nor $c_{j\ell}$ is absorbed by the other, i.e., if $c_{ik} \cdot \overline{c_{j\ell}} \neq 0$ and $\overline{c_{ik}} \cdot c_{j\ell} \neq 0$.*

The chain assignment constraint allows grouping FFs that can be controlled as a shift-register chain under one clock. The serial input to the chain is assumed to come from a PI or a previously controlled FF. The group assignment constraint allows a group of FFs to be controlled from PIs or previously controlled FFs in a non-serial, though not necessarily simultaneous, way.

By the group assignment constraint, G5 and G7 in s27 cannot be assigned the same clock. By the general constraints of the CG, G6 cannot be assigned the same clock as either G5 or G7. Hence, by these rules, the three FFs would be controlled by separate clocks.

In some cases the constraints for group assignment can be relaxed to allow FF_k and FF_ℓ to be assigned to the same clock group if $c_{ik} \neq c_{j\ell}$. As an example, in s27, FFs G5 and G7 can both be assigned to clock 1 giving rise to the following problem. When G1 is set to 1 allowing G2 to control G7 (condition b), condition a also becomes true, enabling G5 to be controlled from G0 regardless of whether G5 is already in its desired state. One solution is to first control G7 by setting G1=1 allowing the complement of G2 to be passed to G7, and then setting $(G1, G3) = (0, 0)$ enabling G5 to be controlled from G0 without affecting the state of G7. Although G5 and G7 can be controlled simultaneously under the correct conditions in the s27 circuit, the process above allows each of the FFs to be controlled independently. After G5 and G7 are assigned to clock 1, we would assign G6 to a second clock.

Consider the procedure to assign FFs to clocks as shown in Figure 5. Beginning with the FFs that are controllable from PIs and scan FFs, we identify the longest chain of FFs that can be grouped according to the chain assignment constraint and insert the chain into the current clock group, C_m. We continue identifying the longest chains that can be assigned to C_m until no additional nodes can be added to the clock group. Note that in adding additional chains to C_m, the nodes of the chain must not only satisfy the chain assignment criteria, but also must satisfy the group assignment criteria with respect to all nodes in C_m.

An alternative to adding maximal length chains to clock groups is to first augment C_m with individual FFs satisfying the group assignment constraint. Subsequently, shift-register chains are constructed from FFs that have been assigned to C_m if they satisfy the chain assignment constraint. In some cases, the trade-off of maximal length chains for greater breadth of individual FFs may allow fewer clocks and/or shorter test lengths.

When no additional FFs can be moved to C_m we 'close' the current clock group and increment m. We continue until all FFs have been assigned to a clock.

4.2 Clock scheduling

The clock assignment procedure generates at most $m + 1$ clock groups; m due to clock control and one possible scan clock. We can use the clock assignments to derive a suitable clock scheduling and to determine the worst case initialization sequence for the clock partitioning. The key to the scheduling strategy is that no FF depends upon a FF in a higher numbered clock group. A FF controlled by clock ϕ_2 can only be affected by FFs in clock group ϕ_1 (or by previous FFs in the same chain in ϕ_2) and will not be affected by FFs in clock groups higher than ϕ_2.

To determine a clock scheduling for an arbitrary state we only need to work backward with respect to the clock groups by initially setting the desired state of the FFs in C_m. We continue working backward towards the FFs in C_1, obtaining a set of test vectors that must be applied to bring the FFs in C_m to their desired state. Next, we set the FFs in C_{m-1} to their desired state and again work backward resulting in a set of test vectors that will bring the FFs in C_{m-1} to their desired state. We continue the process until the FFs in C_1 have been set to their desired state. Reversing the order of the clocks that were enabled within each pass will give a correct clock schedule. In general, as we work backward, we may need to set any other FFs back to their original state if they have been changed.

Using the clock assignment for the s27 example above, assume the current state of s27 is $(G5, G6, G7) = (1, 0, 0)$ and suppose we would like to set s27 to a different state such as (1,1,1). Working backward, G6 must change state so we must *justify* its desired value . To set $G6 = 1$ we must activate ϕ_2 with $G5 = 0$ and $(G1, G3, G7) = (0, 1, 0)$. The current state of G7 satisfies the condition of edge e, but we need to control G5 to 0 before controlling G6. Thus, we activate ϕ_1 to set $G5 = 0$. Summarizing this pass, to control G6 to 1 we enable clocks and apply input vectors according to the following sequence:

	$(G0, G1, G2, G3)$	$(G5, G6, G7)$	
		$(1, 0, 0)$	(current state)
ϕ_1 :	$(0, 0, X, 0)$	$(0, 0, 0)$	
ϕ_2 :	$(X, 0, X, 1)$	$(0, 1, 0)$	

We must still control G5 and G7 to their desired states. In this case we can simultaneously control G5 and G7 by activating ϕ_1 and applying the vector $(G0, G1, G2, G3) = (1, 1, 0, X)$ to reach our desired state $(G5, G6, G7) = (1, 1, 1)$.

4.3 Examples

In this section, we examine four small circuits from the ISCAS-89 benchmark suite; s27, s208, s298, and s382. Table 1 summarizes the results of the circuits under clock control. All results shown use heuristics that primarily minimize the number of clocks used. While none of the circuits shown in the table require scan FFs to control the circuit, we could choose to select one or more scan FFs to reduce the number of clocks as well as the test length.

The maximum test length shown in the table is the worst case test length if all FFs in previous clocks must be changed after bringing the FFs in a higher clock group to their desired state.

Table 1. Clock control results.

			Clock Control		
Circuit	# PIs	# FFs	# Scan FFs	# Clocks	Max. Test Length
s27	4	3	0	2	3
s208	1	8	0	2	28
s298	3	14	0	4	39
s382	3	21	0	3	13

5 Test generation strategies

Test generation algorithms for non-scan and partial-scan circuits require a sequential test generator while those for full-scan require only a combinational test generator. The DFT scheme proposed in this paper shares features from both partial and full scan. As in partial scan, the circuit structure and its S-graph are analyzed to determine the FFs to be controlled. However, the clock control provides an additional dimension of controllability which gives the scheme capabilities resembling full scan. In particular, as in full scan, it is possible to drive the circuit to any desired state using the scan and clock control. The primary differences from full scan are that the state initialization sequence depends on the circuit structure and a complete state observability is not guaranteed.

Our strategy for test generation is to assume, as with full-scan, that the circuit's pseudo-inputs are completely controllable and derive a test for a fault using a combinational test generator. If the fault is detected at a primary output, then we only need to justify the state on the pseudo inputs. Following the strategy used in another sequential test generator [8], we assume a fault-free state justification. Therefore, we can use the procedure described earlier to bring the circuit to the desired state from an unknown (or known) initial state. Hence, no explicit time-frame expansion is necessary. If the fault is detectable only at a pseudo output, we bring around the fault effects to the pseudo inputs in subsequent time frames and use a combinational test generator to propagate the effect to a primary output. Improved results are possible if the test structure is made fault tolerant [3] or a conventional sequential test generator is used.

6 Conclusion

We have presented the main ideas from a work in progress. Our design for testability scheme combines the best features of scan and clock control. We have shown the feasibility of the approach through several examples.

One implementation scheme for clock control appears in the recent paper by Baeg and Rogers [2] and indicates that the hardware cost would be reasonable. These authors group together FFs in strongly connected components in the S-graph. We have proposed a different implementation model in this paper which has a smaller area cost but restricts the clock-control to the activation of a single clock rather than allowing simultaneous activation of a subset of test clocks. Our criterion for grouping FFs is very different from that of Baeg and Rogers.

Our future research includes extending the controllability scheme to achieve full observability.

References

[1] V. D. Agrawal, S. C. Seth, and J. S. Deogun. Design for testability and test generation with two clocks. *Proc. 4th Int'l Symp. on VLSI Design*, pages 112–117, January 1991.

[2] S. Baeg and W. A. Rogers. Hybrid design for testability combining scan and clock line control and method for test generation. *Proc. Int'l. Test Conf.*, pages 340–349, 1994.

[3] S. T. Chakradhar, S. Kanjilal, and V. D. Agrawal. Finite state machine synthesis with fault tolerant test function. *Jour. Electronic Testing: Theory and Applic.*, 4:57–69, February 1993.

[4] K. T. Cheng and V. D. Agrawal. A partial scan method for circuits with feedbacks. *IEEE Trans. on Computers*, C-39(4):544–548, April 1990.

[5] K. T. Cheng and V. D. Agrawal. State assignment for testable design. *Int'l. Jour. Computer Aided Design*, 3:291–307, 1991.

[6] K. L. Einspahr, S. C. Seth, and V. D. Agrawal. Clock partitioning for testability. *Proc. 3rd IEEE Great Lakes Symp. on VLSI Design*, pages 42–46, March 1993.

[7] W.-C. Fang and S. K. Gupta. Clock grouping: A low cost DFT methodology for delay testing. *Proc. Design Automation Conf.*, pages 94–99, 1994.

[8] A. Ghosh, S. Devadas, and A. R. Newton. Test generation and verification for highly sequential circuits. *IEEE Trans. Computer Aided Design*, 10:652–667, May 1991.

[9] K. B. Rajan, D. E. Long, and M. Abramovici. Increasing testability by clock transformation (getting rid of those darn states). *Proc. 14th IEEE VLSI Test Symp.*, Apr/May 1996.