1999

A Synthesis for Testability Scheme for Finite State Machines Using Clock Control

Kent L. Einspahr
Concordia University, Seward, NE, Kent.Einspahr@cune.edu

Shashank K. Mehta
University of Pune, India

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

A Synthesis for Testability Scheme for Finite State Machines Using Clock Control

Kent L. Einspahr, Shashank K. Mehta, and Sharad C. Seth

***Abstract*—A new method is proposed for improving the testability of a finite state machine (FSM) during its synthesis. The method exploits clock control to enhance the controllability and observability of machine states. With clock control it is possible to add new state transitions during testing. Therefore, it is easier to navigate between states in the resulting test machine. Unlike prior work, where clock control is added to the circuit as a post-design step, here, clock control is considered in conjunction with a symbolic scheme for encoding the states of the FSM. The encoding is shown to result in significant reductions in the interstate distances in the benchmark FSM's. Further, the observability of the encoded states can be improved by adding two primary outputs to the circuit such that a fixed input sequence forms a distinguishing sequence for all states. Theoretical results show that for a large class of FSM's, the testability improvements are comparable to those achievable by scan designs. Experimental results show that available test pattern generation tools are able to take advantage of the enhanced testability in producing shorter test sequences, particularly for machines with poor connectivity of states.**

***Index Terms*— Automatic test pattern generation (ATPG), design-for-testability (DFT), test synthesis, testing**

I. Introduction

THIS paper proposes a new technique for improving the testability of a finite state machine (FSM) during its synthesis. The technique exploits the clock-control concept [2] to enhance the controllability and observability of machine states. With clock control it is possible to augment the original state transition graph (STG) with new transitions during testing. The resulting *test machine* can, therefore, have a better-connected STG, making it easier to navigate between states. Prior work on clock control has mainly been concentrated on using it as a design-for-testability (DFT) tool [3], [7]–[9], [17]. Here, clock control is employed in conjunction with a symbolic state encoding scheme (called *split-coding*) to serve as a synthesis-for-testability (SFT) method. The synthesized circuits are shown to have shorter test sequences than the circuits synthesized without clock control.

Furthermore, these gains are achievable by standard sequential automatic test pattern generation (ATPG) tools. In the proposed scheme, the test machine embeds the STG of the original FSM. The embedding concept has appeared before (see [4]–[6], and [11]) but has not been explored in the context of state assignment and clock control. The distances between the states can be reduced by adding new transitions between distant states [11], [17], but this approach may entail significant logic overhead. Instead, we propose to effect new transitions by simply disabling the clock to a group of flip-flops (FF's) in the circuit. The resulting scheme affords state controllability comparable to scan designs. The idea of disabling the clock has been used before in BALLAST [16] which is a DFT scheme applicable to pipelined circuits with limited feedback and feedforward connections. In contrast, the work proposed here represents a SFT technique for circuits with behavior described by FSM's.

A potential advantage of the proposed scheme over scan is that efficient navigation between states is possible without entering illegal (nonfunctional) states. Test pattern generators may exploit this feature by discarding the faults that require entering an illegal (nonfunctional) state for their detection as such faults are functionally-redundant.

The state observability is improved by judiciously adding two primary outputs to the circuit such that a fixed input sequence forms a distinguishing sequence (DS) for all (split-coded) states. The DS is of length $O(\log N)$[1] where N is the number of valid states, and returns the good machine to the state at the beginning of the sequence. The latter property can be very helpful in reducing the test length in circuits with poor observability.

The proposed technique can be used directly to enhance the testability of FSM controllers with and without data paths. Usually, data paths have good controllability and observability and are easier to test than controllers [13]. Therefore, the focus here on simplifying test generation for a FSM controller is justified.

Preliminary ideas related to this work were reported earlier [14]. The original approach has been substantially extended to apply to arbitrary FSM's and extensive experimental results have been added in support of its usefulness.

The rest of the paper is organized as follows. Section II provides the background on clock control to enhance state controllability. This is followed in Section III by a formal definition of split-codes that form the basis for the state assignment using clock control. Several properties of the split-codes relevant to our work are also discussed. Section IV shows how split-codes can be used in state assignment so as to

Manuscript received December 23, 1998; revised June 10, 1999. This paper was recommended by Associate Editor L. Stock.

K. L. Einspahr is with the Department of Computer Science, Concordia University, Seward, NE 68434-1599 USA.

S. K. Mehta is with the University of Pune, Pune, 411007 India.

S. C. Seth is with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588-0115 USA.

Publisher Item Identifier S 0278-0070(99)09936-4.

[1] In this paper, $\log N$ will always be assumed to be $\log_2 N$.

improve navigation between states through clock control. Two schemes for implementing the clock control are also described in this section. For some circuits with poor observability it may not be enough to improve controllability of states, therefore, Section V proposes a scheme to enhance the state observability. Section VI provides experimental data in support of the theoretical results in Sections IV and V and concluding remarks appear in Section VII.

II. State Controllability and Two Clock Approach

Time-frame-based test generation is an iterative process in which, after an uncovered fault is chosen as target, the test generator passes through two phases: the *state control* phase wherein a search is made for an input sequence that takes the circuit to a state in which the fault can be excited, and the *state observation* phase wherein a search is made for an input sequence which propagates the fault effect to a primary output. The state controllability of a FSM refers to the ease with which the test generator can navigate to the desired state. The better the connectivity of the STG, the easier is the navigation. Thus, the controllability of a FSM can be improved by adding new transitions to its STG during testing. The well-known DFT methods, such as scan and partial-scan, can be viewed as improving controllability by improving the connectivity of the STG. Other suggestions along the same line include: adding new transitions to the STG [1], [10], [12], adding new (illegal) states and transitions to the STG [15], adding reset to a "central" state to minimize interstate distances [11], and using clock control [2], [8], [14]. The clock control technique is further elaborated in this section as it forms the basis for this paper. We shall return to the issue of state observability in Section V.

A. Clock Control

The basic idea of clock control is to divide the FF's in the circuit into two or more groups and disable the clock to some groups of FF's during testing. For simplicity of design and test application, a fixed grouping of FF's is preferable. The normal circuit behavior is realized by enabling the clock to all FF groups. However, during testing, new transitions may be realized because the states of the FF's in the disabled group(s) cannot change. As an example, suppose that the FF's are divided into two groups α and β and that it is possible to independently control the clock signals, ϕ_α and ϕ_β, to the two groups. Fig. 1(a) schematically shows how new transitions may result from a normal transition because of the clock control. Assume that the assigned state $\langle\alpha_1, \beta_1\rangle$ changes to the assigned state $\langle\alpha_2, \beta_2\rangle$ under some input in the original FSM, where α_i and β_i denote the values of the corresponding group of FF's. When ϕ_α (ϕ_β) is disabled in a test mode, the new state under the same initial state and input would be $\langle\alpha_1, \beta_2\rangle$ ($\langle\alpha_2, \beta_1\rangle$). When the clock to both groups is disabled, the current state of the circuit will be held. Because the clock to each group is being independently controlled, two binary control signals are required in this example.

In the general case, a circuit with l FF groups will have 2^l clock-control modes, which is the number of subsets of the

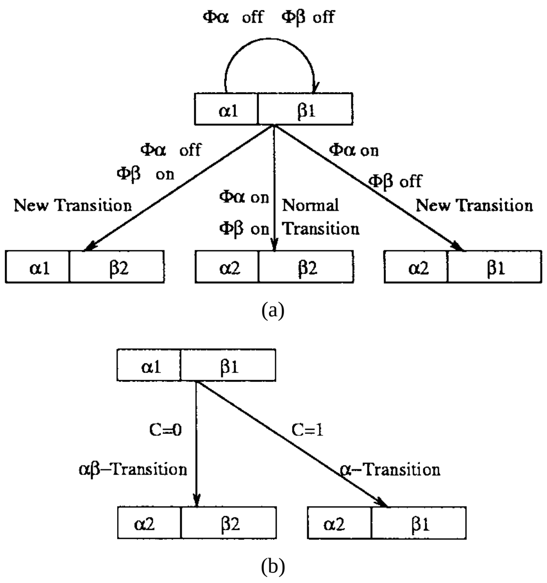


Fig. 1. State transitions under clock control.

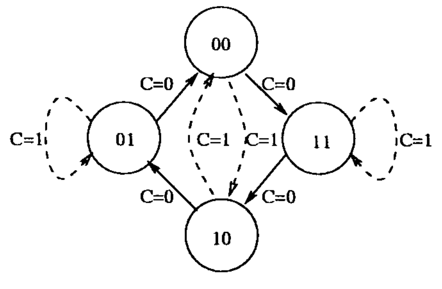
groups that may be turned off. Thus, each transition in the original FSM generates a spectrum of 2^l transitions, one for every clock-control mode. The circuit will require l additional controls, one for each FF group. In this paper we restrict discussion to just two FF groups and assume the clock control of only one group. Specifically, it is assumed that a control signal, C, *either enables* ($C = 0$) *or disables* ($C = 1$) *the clock to the* β group of FF's; the clock to the α group of FF's is assumed to always be active. The transitions in the two clock modes ($C = 0$ and $C = 1$) will be denoted, respectively, as $\alpha\beta$-transitions and α-transitions, as shown in Fig. 1(b). Both of these modes will be assumed to be available during testing and the term *test machine* will be used to refer to the FSM which includes both the α and $\alpha\beta$ transitions in its STG. *Normal machine* or *original machine* will refer to the STG consisting only of the original ($\alpha\beta$) transitions.

Fig. 2(a) shows an example STG of a modulo-4 counter under two-clock control. Here, the first bit corresponds to the α group and the second bit corresponds to the β group. The $\alpha\beta$ transitions ($C = 0$) corresponding to the normal counter function are shown in solid and the α-transitions ($C = 1$) are shown by broken lines. Because of the addition of the latter transitions, the worst-case distance between two states is reduced from 3.0 to 2.0 and the average distance between two states is reduced from 1.5 to 1.125.

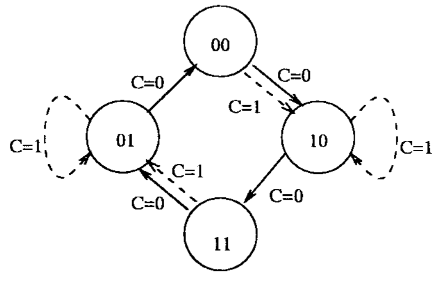
B. The Role of State Assignment Versus FF Partitioning

Consider again the example of the modulo-4 counter implemented with two-clock control, but this time with the state assignment shown in Fig. 2(b). In this case, the test machine turns out to have the same maximum and average interstate distances as the original counter, i.e., 3.0 and 1.5, respectively. This shows that the effectiveness of the two-clock approach not only depends on the FF partitioning but also on the state assignment.

FF partitioning by itself is a design-for-testability scheme which, like scan, can be applied to an existing design, i.e.,



(a)



(b)

Fig. 2. Test machine of modulo-4 counter.

after the design and synthesis have taken place. Alternatively, clock control can be considered at the time of state assignment as a SFT scheme. In the latter case, the state assignment as well as the FF partitioning can be optimally determined for maximum reduction in the interstate distances. In the case of n FF's there are 2^{n-1} ways to partition the FF's. But the presynthesis options are much larger in number. There are $2^n!$ ways to assign a binary code to the states (assuming 2^n states) and for each state assignment there are 2^{n-1} ways to partition the FF's. It is natural to ask whether the choices under the DFT scheme (FF partitioning only) are sufficient to improve the controllability. Although it is difficult to answer this question, the following examples show that for some state assignments only negligible improvement is possible, no matter how the FF's are partitioned.

Consider a modulo-N $(N = 2^n)$ counter in which the states are connected in the cycle: $S_0, S_1, \ldots, S_{N-1}, S_0$. The average distance between two states in the original counter is $(N-1)/2$. For the first example, assume that successive states of the cycle are assigned the successive Gray codes. Irrespective of how the FF's are partitioned, either the α-bits or the β-bits of state S_j and $S_{j+1 (\mathrm{mod}\ 2^n)}$ will be the same for all j because successive Gray codes differ in only one bit. The α-transition will either loop back to the initial state of the transition or coincide with the corresponding $\alpha\beta$-transition [see Fig. 2(b)]. Thus, the distances between the states in the original STG and the test machine will be identical.

For the second example, consider the binary number encoding in the modulo-N counter, i.e., the binary code of number i is assigned to state S_i with FF's ordered as $f_{n-1} \cdots f_0$. If we are restricted to partitions α: $f_{n-1} \cdots f_p$ and β: $f_{p-1} \cdots f_0$ for an arbitrary p, then the average distance cannot be reduced to less than $3N/8$ in the test machine. For proof see Observation 2 in the Appendix. This reduction can be considered negligible compared to that due to *scanning* where the distance between the states in the test machine is $\lceil \log N \rceil$. In this paper, we present a state encoding called *split-coding* which ensures $O(\log\ N)$ distance between the states in the test machine for a large class of circuits synthesized with two clocks.

III. Split-Codes

In order to motivate the definition of split-codes, we make certain simplifying assumptions which will be relaxed later. Assume the FSM under consideration has $N = m \cdot 2^k$ states, where m and k are integers, $m \geq k$, and the states of the machine are connected in a Hamiltonian cycle (HC) of transitions. The objective of split-coding is to allow the implementation of a test machine with desirable state controllability. If we squeeze the successive m states in the cycle into *super nodes*, then the test machine defined by the split-code will reduce to the *barrel shifter* network. In this network all the super nodes are in a directed cycle and further, each super node is directly connected to the super nodes at distance 2^k for all k. In this network, the maximum distance between nodes is $k \leq \log\ N$ and within a super node, the split-code ensures that the maximum distance between states is m which can be chosen to be $O(\log\ N)$. Fig. 3 (presented later in this section) illustrates the super nodes SN0, ..., SN3 for a specific example discussed below.

A *split-code*, $\mathcal{S}(m,k)$, is defined for any pair of integers: $0 < k \leq m$. It is a sequence of pairs of integers $\langle \alpha^0, \beta^0 \rangle$, $\langle \alpha^1, \beta^1 \rangle$, $\langle \alpha^2, \beta^2 \rangle, \ldots$, where $0 \leq \alpha^j \leq m-1$ and $0 \leq \beta^j \leq 2^k - 1$, defined recursively by

1) $\langle \alpha^0, \beta^0 \rangle = \langle 0, 0 \rangle$;
2) $\langle \alpha^{j+1}, \beta^{j+1} \rangle = \langle \alpha^j + 1 (\mathrm{mod}\ m), \beta^j + 2^{\alpha^j} (\mathrm{mod}\ 2^k) \rangle$, for all $j \geq 0$.

Simplification of the expression for $\langle \alpha^{m \cdot 2^k}, \beta^{m \cdot 2^k} \rangle$ leads to the equation (proved as Observation 4 in the Appendix)

$$\langle \alpha^{m \cdot 2^k}, \beta^{m \cdot 2^k} \rangle = \langle \alpha^0, \beta^0 \rangle.$$

This observation establishes that $\langle \alpha^j, \beta^j \rangle = \langle \alpha^{j (\mathrm{mod}\ m \cdot 2^k)}, \beta^{j (\mathrm{mod}\ m \cdot 2^k)} \rangle$ so there are only $m \cdot 2^k$ distinct pairs, namely, $\langle \alpha^0, \beta^0 \rangle$, $\langle \alpha^1, \beta^1 \rangle$, $\ldots, \langle \alpha^{m \cdot 2^k - 1}, \beta^{m \cdot 2^k - 1} \rangle$. Table I shows the elements of $\mathcal{S}(3,2)$. Refer to the Appendix for a rigorous definition and properties of split-codes.

For notational simplicity, arithmetic operations on the values of α and β will be understood to be modulo-m and modulo-2^k, respectively. Similarly, arithmetic operations on the indexes of $\mathcal{S}(m,k)$ will be understood to be modulo-$m \cdot 2^k$.

A. A Property of Sequences

We present a result on the split-code sequences which will be used to show the $O(\log\ N)$ bound on the distances between the states in the test mode for a large class of circuits when synthesized with two clocks. For convenience, we first define the notions of *unit-step* and *unit-step sequence*.

Definition: The following pairs of code-pairs, for all j, will be called *unit-steps*:

1) $\alpha\beta$-step: $(\langle \alpha^j, \beta^j \rangle, \langle \alpha^{j+1}, \beta^{j+1} \rangle)$;
2) α-step: $(\langle \alpha^j, \beta^j \rangle, \langle \alpha^{j+1}, \beta^j \rangle)$.

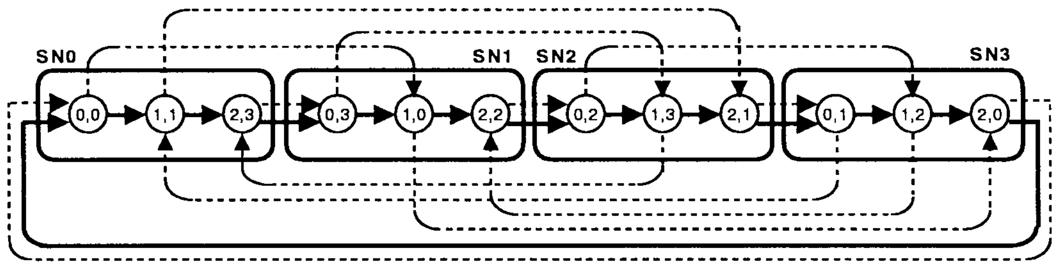


Fig. 3. $\alpha\beta$ (solid) and α (dashed) transitions for $\mathcal{S}(3,2)$.

TABLE I
THE SPLIT-CODE FOR $m = 3, k = 2$

Index	0	1	2	3	4	5	6	7	8	9	10	11
Split-Code	$\langle 0,0\rangle$	$\langle 1,1\rangle$	$\langle 2,3\rangle$	$\langle 0,3\rangle$	$\langle 1,0\rangle$	$\langle 2,2\rangle$	$\langle 0,2\rangle$	$\langle 1,3\rangle$	$\langle 2,1\rangle$	$\langle 0,1\rangle$	$\langle 1,2\rangle$	$\langle 2,0\rangle$

Definition: A sequence $\langle\alpha_0,\beta_0\rangle$, $\langle\alpha_1,\beta_1\rangle,\ldots,\langle\alpha_p,\beta_p\rangle$ will be called a *unit-step* sequence if each pair of successive elements, $(\langle\alpha_j,\beta_j\rangle, \langle\alpha_{j+1},\beta_{j+1}\rangle)$, is a unit-step ($\alpha\beta$-step or α-*step*).

Theorem 1: For any sequence of successive code-pairs of $\mathcal{S}(m,k)$

$$S_{i,j}\colon \langle\alpha^i,\beta^i\rangle,\langle\alpha^{i+1},\beta^{i+1}\rangle,\ldots,\langle\alpha^{i+q},\beta^{i+q}\rangle(=\langle\alpha^j,\beta^j\rangle)$$

there exists a unit-step sequence

$$\langle\alpha^i,\beta^i\rangle,\langle\alpha^{i+n_1},\beta^{i+n_1}\rangle,\langle\alpha^{i+n_1+n_2},\beta^{i+n_1+n_2}\rangle,\\ \ldots,\langle\alpha^{i+n_1+n_2+\cdots+n_p},\beta^{i+n_1+n_2+\cdots+n_p}\rangle(=\langle\alpha^j,\beta^j\rangle)$$

with $p < 2m$. Further, this sequence is contained in $S_{i,j}$.

The desired sequence is constructed in two steps. First, an $\alpha\beta$-step sequence is constructed from $\langle\alpha^i,\beta^i\rangle$ to $\langle\alpha^l,\beta^l\rangle$ which is the first code starting from $\langle\alpha^i,\beta^i\rangle$ such that $\alpha^j = \alpha^l$. Second, a unit-step sequence of length m is constructed which transforms β^l to β^j. Since the length of the first subsequence cannot exceed $m-1$, the total length is bounded by $2m-1$. Refer to the Appendix for a proof.

Example: For the split-code $\mathcal{S}(3,2)$, shown in Table I, let $i = 6$ and $j = 4$. A possible unit-step sequence is

$$\langle\alpha^6,\beta^6\rangle = \langle 0,2\rangle,\langle 1,3\rangle,\langle 2,3\rangle,\langle 0,3\rangle,\langle 1,0\rangle = \langle\alpha^4,\beta^4\rangle.$$

The successive steps in this unit-step sequence are $\alpha\beta$, α, α, and $\alpha\beta$.

Fig. 3 illustrates possible $\alpha\beta$- and α-steps for $\mathcal{S}(3,2)$ by the solid and dashed edges, respectively. The super node concept described earlier is also shown by grouping the successive $m = 3$ states.

B. Parameters m and k

In Section IV we illustrate how binary codes are generated for the states of a FSM using a split-code. As preparation toward that end, we discuss here how the parameters of the split-code are selected. Since the total number of pairs in $\mathcal{S}(m,k)$ is $m\cdot 2^k$, it is necessary that this number is at least as large as the number of states. Thus, two necessary conditions for any pair of parameters are as follows:

C1) $k \le m$ (by definition of $\mathcal{S}(m,k)$);

TABLE II
OPTIMUM PARAMETERS FOR $N \le 3072$, WITH $m = \max\{k, \lceil N/2^k\rceil\}$

Range	of	N	k
2	–	6	1
7	–	20	2
21	–	48	3
49	–	112	4
113	–	288	5
289	–	640	6
641	–	1408	7
1409	–	3072	8

C2) $\log N \le k + \log m$, where N denotes the number of states.

We shall also see in the next section that the choice of m will determine the bound for the interstate distance in the test machine. Thus two desirable conditions for the parameters to satisfy are as follows:

D1) $\lceil\log N\rceil = k + \lceil\log m\rceil$, i.e., the split-code-based assignment could be done with $\lceil\log N\rceil$ bits;

D2) $m \le \lceil\log N\rceil$.

The following observation shows that, for any integer N, parameters can be chosen that satisfy conditions C and at least one of the conditions D.

C. Observation 1

1) For any positive integer N, there exist integers m and k satisfying $N \le m\cdot 2^k$, such that:
   a) $k \le m \le 2\cdot\lceil\log N\rceil$;
   b) $k + \lceil\log m\rceil = \lceil\log N\rceil$.
2) For any positive integer N, there exist integers m and k satisfying $N \le m\cdot 2^k$, such that:
   a) $k \le m \le \lceil\log N\rceil$;
   b) $k + \lceil\log m\rceil \le \lceil\log N\rceil + 1$.

It is interesting to note that, for $N \le 3072$, optimum parameters satisfy conditions D1 and D2. Table II shows how to determine the optimum values of the parameters for $2 \le N \le 3072$. For example, if $N = 185$, then $k = 5$ (from the table) and $m = \max\{k, \lceil N/2^k\rceil\} = \max\{5, \lceil 185/32\rceil\} = \max\{5, 6\} = 6$. This choice of parameters ensures that split-code-based encoding can be accomplished with a minimum

TABLE III
STATE ASSIGNMENT FOR MODULO-10 COUNTER USING SPLIT-CODING

State	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Split-Code	$\langle 0,0\rangle$	$\langle 1,1\rangle$	$\langle 2,3\rangle$	$\langle 0,3\rangle$	$\langle 1,0\rangle$	$\langle 2,2\rangle$	$\langle 0,2\rangle$	$\langle 1,3\rangle$	$\langle 2,1\rangle$	$\langle 0,1\rangle$

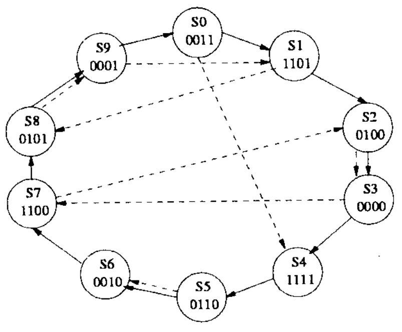


Fig. 4. Modulo-10 counter: split-code-based state assignment with added transitions shown by broken lines.

possible number of bits because condition D1 is satisfied as

$$\lceil \log N \rceil = 8 = 5 + 3 = k + \lceil \log 6 \rceil = k + \lceil \log m \rceil.$$

Also, condition D2 is satisfied as follows:

$$m = 6 < 8 = \lceil \log 185 \rceil = \lceil \log N \rceil.$$

IV. SYNTHESIS WITH SPLIT-CODES

In this section we describe the split-code assignment to the states of an arbitrary STG, reducing the distances between the states in the test machine and improving the state controllability. We begin with an example of state assignment for a modulo-10 counter using the split-code $\mathcal{S}(3, 2)$ (see Table I). Subsequently, we address the state assignment problem for arbitrary FSM's.

A. Example of State Assignment with Split-Codes

Split-code state encoding is performed in two steps. In the first step, the split-code code-pairs are assigned to the successive states in the cycle as shown in Table III. In the second step a binary assignment is generated by arbitrarily assigning binary codes to α's and β's. Fig. 4 illustrates the binary state assignment to the modulo-10 counter when α values are encoded as 0:00, 1:11, and 2:01 and β values are encoded as 0:11, 1:01, 2:10, and 3:00. The figure shows the STG of the test machine where solid arrows represent $\alpha\beta$-transitions and broken arrows represent α-transitions. The maximum and the average distances between the states in the normal machine (solid arrows only) are 9.0 and 4.5, respectively; the distances reduce to 6.0 and 2.7, respectively, when the α-transitions are added. Observe that the new transitions depend only upon the split-code assignment to the states, i.e., the first step. Thus, the distances between the states in the test machine are independent of the choice of binary encoding of the α and β components, although this choice can affect the complexity of the synthesized logic.

B. State Assignment

In the example above we observe that the topology of the test machine depends only upon the normal machine topology and the split-code assignment; it does not depend on the binary encoding of α and β components. To understand how the split-code pairs are assigned to the states, let us first assume that the STG has a HC: $S_0, S_1, \ldots, S_{N-1}$. Let us further assume that $N = m \cdot 2^k$ for some suitable values of the parameters. If each state S_i is assigned the code-pair $\langle \alpha^i, \beta^i \rangle$, then every $\alpha\beta$-step coincides with a transition in the normal machine. Thus, every unit-step coincides with a transition in the test machine. Consequently, every unit-step sequence is a path in the test machine. From Theorem 1 and Observation 1, there must exist a path of length $O(\log N)$ between any pair of states. This observation regarding the ideal case leads to the following criterion for the state assignment in the general case: The number of $\alpha\beta$-steps that do not coincide with the transitions in the normal machine should be minimum.

A formal description of the state assignment algorithm is as follows.

1) Find a minimum nonintersecting *path-cover*, i.e., a set of nonintersecting paths, $P^0 = S_0, S_1, \ldots, S_{N_0-1}$, $P^1 = S_{N_0}, \ldots, S_{N_0+N_1-1}$, etc., covering all states in the STG of the given FSM.
2) Determine optimum values of parameters m and k from Table II.
3) Assign code-pair $\langle \alpha^j, \beta^j \rangle$ to state S_j for all j.
4) Independently assign binary codes to the elements of the α and β fields, α: $\{0, 1, \ldots, m-1\}$ and β: $\{0, 1, \ldots, 2^k - 1\}$, respectively.[2]

This method of state assignment results in the minimum number of $\alpha\beta$-steps that do not coincide with transitions in the normal machine because the number of such steps is equal to the number of paths in the path-cover.

C. State Controllability

One of the consequences of the split-code assignment algorithm given above is that each path, P^l, used in the assignment is a sequence of nodes with successive code-pairs. Application of Theorem 1 to any path of the cover readily leads to the following result.

Theorem 2: In a sequential circuit synthesized as described above, if states S_i and S_j belong to the same path P^l (of the path-cover) with $i < j$, then in the test machine there exists a path from S_i to S_j of length no more than $2m - 1$. Further, this path is confined to the states of P^l.

Proof: By construction, all $\alpha\beta$-steps coincide with transitions of the original STG, except the $\alpha\beta$-steps

[2] A state assignment method, such as [18], that attempts to optimize logic can be developed for this step. However, in this paper arbitrary binary codes are assigned to the elements of the α and β fields.

$(\langle\alpha^{N_0+\cdots+N_p},\beta^{N_0+\cdots+N_p}\rangle,\ \langle\alpha^{N_0+\cdots+N_p+1},\beta^{N_0+\cdots+N_p+1}\rangle)$ for all p which are associated with the end nodes of the paths. Consequently, the corresponding α-steps also coincide with transitions in the test machine. Thus, the unit-step sequence promised in Theorem 1 is a path in the test machine. ●

Recall Observation 1 in Section III which ensures that the parameters can be chosen so that m is less than or equal to $\lceil\log N\rceil$. Therefore, the length of the path from S_i to S_j will be bounded by $2\lceil\log N\rceil - 1$. This result has a significant implication for the FSM's which have a HC in their STG. It is noteworthy that of the 53 MCNC benchmarks, 16 STG's have a HC and, in the remaining STG's, only a very small number of additional edges need to be added to achieve the same property [14].

Corollary: If the path cover is a HC, then there exists a path from any state S_i to any state S_j of length no more than $4m-1$ in the test mode. Furthermore, this path does not leave the functional states of the FSM. The average length of the path between the states will be $2.25m$.

Proof: From Theorem 2, if $i < j$, then there is a path of length no more than $2m - 1$. If $j < i$, then the path is constructed in three steps because $(\langle\alpha^{N-1},\beta^{N-1}\rangle, \langle\alpha^0,\beta^0\rangle)$ is not a unit step (unless $N = m \cdot 2^k$). The path is the concatenation of the following:

1) a path from S_i to S_{N-1} of length less than $2m$ (assured by the theorem);
2) a transition from S_{N-1} to S_0 (since it is a HC);
3) a path from S_0 to S_j which is less than $2m$ in length (from Theorem 2).

Thus, the total length of the path is less than $4m$. ●

D. Implementation of Two-Clock Control

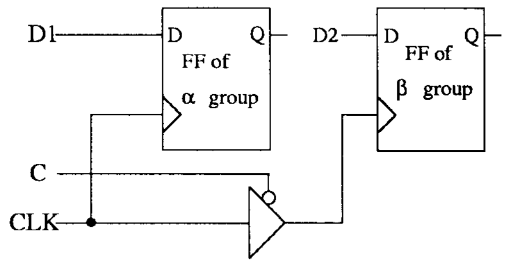
As mentioned in Section II, a new input, C, is required to enable or disable the clock to the β FF's. Two possible implementations of this function are as follows.

The first approach is direct and affects the clock-distribution logic. In this approach, the input C is used to control a tri-state buffer along the path from the clock source to each FF in the β group, see Fig. 5(a). No timing penalty and only a small area penalty will be involved in such an implementation. However, because of its impact on the clock-tree design, the approach may not be readily acceptable to the designers.

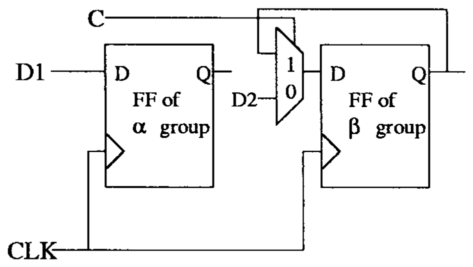
The second way of implementing the clock control does not affect clock distribution. Instead, the control signal C is connected as the select input of a two-input mux feeding each FF in the β group [see Fig. 5(b)]. When $C = 0$, the normal data input is steered to the FF input; when $C = 1$, the FF state is recirculated through the mux. The overhead of this scheme is somewhat similar to scan, however, there may be less area/time penalty because, unlike scan, one of the inputs to the mux is internal to the mux-FF combination.

V. Observability

The preceding section was devoted to the synthesis of circuits using split-coding to improve the state controllability during testing. In this section we develop a method to improve the observability of a circuit which is synthesized using a split-code. As in the case of state controllability, the method is ideal for FSM's with HC but is also quite effective, in general.



(a)



(b)

Fig. 5. Implementations of clock control.

In Section II we observed that in the second phase of each iteration a test generator tries to propagate the fault effect to a primary output. If the fault effect cannot be propagated to the output in the current time frame, it may be necessary to propagate the fault effect to one or more secondary outputs in the current time frame and propagate the effect from the state line(s) to a primary output in subsequent time frames. In general, the second phase requires a search for an input vector sequence which can distinguish (through the output sequence) between the target state in the good and the faulty machines. An input sequence which can identify a state from all other states is called a *distinguishing sequence* (DS). A circuit will have high state observability if it has a short DS for all of its states.

The observability of a circuit does not depend on the state assignment during synthesis; rather, the DS is a characteristic of the underlying STG. There are two ways to improve the state observability of a FSM: 1) add new transitions between the states of the STG with carefully chosen outputs, and 2) expand the observation space, i.e., add new output bits and prudently choose their values for every transition (Mealy outputs are assumed here). The former is possible only when there are input patterns not used in the specification of the FSM. In this paper we focus only on the second option.

The objective of improving the state observability is to determine the state identity using a short DS while incurring minimal additional hardware cost. We present a method to achieve this goal by adding two new primary outputs, P_1 and P_2 (see Fig. 6). As suggested in [15] and elsewhere, these outputs may be multiplexed with other available outputs without incurring any pin overhead. In synthesizing the circuit with a split-code, the DS should generate enough information to identify the α and β values of the code assigned to the

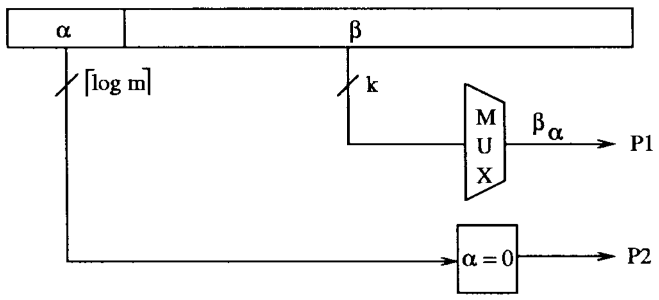


Fig. 6. Logic to enhance observability.

initial state. To do this, we define $P_1 = \beta_\alpha$ (the αth bit of β) and $P_2 = (\alpha = 0)$ (i.e., $P_2 = 1$ when $\alpha = 0$).

As discussed in Section IV, when the normal machine of the FSM has a HC and $N = m \cdot 2^k$, all unit steps are transitions in the test machine. Consider m successive α-step transitions from an arbitrary state with the code $\langle \alpha, \beta \rangle$, where values of P_1 and P_2 for each transition are shown above the arrow

$$\langle \alpha, \beta \rangle, \xrightarrow{\beta_\alpha, 0} \langle \alpha + 1, \beta \rangle, \xrightarrow{\beta_{\alpha+1}, 0} \cdots, \langle m-1, \beta \rangle, \xrightarrow{\beta_{m-1}, 0} \langle 0, \beta \rangle, \xrightarrow{\beta_0, 1} \langle 1, \beta \rangle, \xrightarrow{\beta_1, 0} \cdots, \langle \alpha - 1, \beta \rangle, \xrightarrow{\beta_{\alpha-1}, 0} \langle \alpha, \beta \rangle.$$

Observe that the value of α is $m - j$ where $P_2 = 1$ occurs at the jth position in the sequence. Also note that the sequence of P_1 values is the result of $m - \alpha$ right circular shifts of β-reverse (i.e., $\beta_0 \beta_1 \cdots \beta_{k-1}$). Therefore, these outputs uniquely identify the initial state leading to the conclusion that m successive α-step transitions constitute the DS for any state. For example, in the modulo-12 counter encoded with $\mathcal{S}(3,2)$, the output of the DS from state $\langle 1, 3 \rangle$ is

$$\langle 1, 3 \rangle, \xrightarrow{1,0} \langle 2, 3 \rangle, \xrightarrow{0,0} \langle 0, 3 \rangle, \xrightarrow{1,1} \langle 1, 3 \rangle.$$

In this example $P_2 = 1$ occurs at step two (counting from zero), i.e., $j = 2$. Thus, $\alpha = m - j = 3 - 2 = 1$. Further, β-reverse can be obtained by $m - \alpha = 2$ left circular shifts on the output sequence of P_1, i.e., 101. So β-reverse $= 110$, and $\beta = 011 = 3$. Therefore, the initial state is correctly determined to be $\langle 1, 3 \rangle$.

The advantage of this DS is that its length is m (approximately $\log N$) for all states. In other words, it is short and identical for all states. Furthermore, the characteristic that it returns to the initial state could be useful for some applications. Although this sequence is a DS only in ideal circuits (with HC and $N = m \cdot 2^k$), it provides significantly improved state observability in the general case, as we shall see in the next section.

VI. Empirical Evaluation of the Method

To evaluate the impact of the suggested method of circuit synthesis on testing, two experiments were performed on the MCNC benchmark FSM's. The objective of the first experiment was to compare the shortest distances between states in the STG's of the original and the test machines. The change in shortest distances was expected to provide a measure of the improvement in state controllability, independent of a specific automatic test pattern generation (ATPG) system. The second experiment involved the use of existing ATPG programs on circuits synthesized with and without testability enhancements. The objective was to determine the extent to which such programs can exploit the improved state controllability and observability to produce shorter test sequences for clock-controlled circuits.

The 53 MCNC FSM benchmarks were considered for the experiments. Of these, ten were eliminated for the following reasons: four FSM's had a dead state (*ex2, ex3, ex5, ex7*), three (*bbsse, scf, sse*) had more than one unreachable state (if there was only one such state it was turned into an explicit reset state), two FSM's (*donfile, s1a*) reduced to combinational circuits when synthesized, and *planet1* was indistinguishable from *planet*. Experiments were done with the remaining 43 FSM's.

As discussed in Section IV, split-code code-pairs were assigned to states by computing a path cover for the states. First, a HC was computed in the FSM by temporarily adding the fewest possible edges; the added edges were subsequently dropped after this step. Starting at a randomly selected state, successive split-code code pairs were assigned to the states in the order of their appearance in the HC.

Path covers were generated for all 43 FSM's tested. The following list summarizes the results in terms of the number of benchmarks and, within parentheses, the number of edges required to form a HC: 13 (0), 12 (1), 4 (2), 6 (3), 4 (4), and 4(>4). The list shows that in less than 10% of the cases the path cover required more than four paths. Further detailed analysis of the results shows no apparent relationship between the percentage of the extra edges required and the number of machine states.

A. Shortest Distance Analysis

The shortest distance was computed between every pair of distinct states in the normal machines and the test machines with the assumption that the shortest distance between a state and itself is zero. Table IV shows the average and the maximum shortest-distances between the states in the STG's of the original machine and the test machine (i.e., the machine with α transitions). The numbers within parentheses express the percentage reduction (Red) in the distance. For reference, the number of states and transitions are also shown for each original FSM.

The magnitude of reduction in the distances between the states should depend on the connectivity of the original FSM. Greater improvement should be expected in graphs with poorer connectivity. In the directed cycles (which are the strongly connected graphs with poorest connectivity), if one outbound edge can be added from each state to any other state, then the average case shortest-distance between the states can be reduced, at best, to $O(\log N)$, where N is the number of states in the FSM. To put the results in this context we also show the average distance as a multiple of $\log N$

$$\textit{normalized average distance} = \textit{Average Dist}/\lceil \log N \rceil.$$

We only report the results for FSM's with greater than 14 states because the results for small FSM's are difficult to interpret.

TABLE IV
COMPARISON OF SHORTEST DISTANCES

FSM	States	Trans	Maximum Distance		Average Distance		Norm Average Dist	
			Original	Enhanced(Red)	Original	Enhanced(Red)	Original	Enhanced
s510	47	77	46	7 (84%)	15.62	3.97 (74%)	2.60	0.66
planet	48	115	24	10 (58%)	9.68	4.18 (56%)	1.61	0.70
kirkman	16	385	15	5 (66%)	5.67	2.66 (53%)	1.42	0.66
s1494	48	250	22	6 (72%)	8.19	3.34 (59%)	1.37	0.56
s1488	48	251	22	7 (68%)	8.19	3.45 (57%)	1.37	0.57
sand	32	184	22	6 (72%)	6.56	3.07 (53%)	1.31	0.61
s420	18	137	17	5 (70%)	5.84	2.52 (56%)	1.17	0.50
s208	18	153	17	5 (70%)	5.84	2.43 (58%)	1.17	0.49
s298	218	1096	19	5 (73%)	7.69	3.09 (59%)	0.96	0.39
tma	20	44	10	6 (40%)	4.75	2.89 (39%)	0.95	0.58
styr	30	166	11	6 (45%)	4.73	2.72 (42%)	0.95	0.54
s832	25	245	11	7 (36%)	4.34	2.80 (35%)	0.87	0.56
s820	25	232	11	6 (45%)	4.34	2.58 (40%)	0.87	0.52
pma	24	73	10	5 (50%)	4.31	2.55 (40%)	0.86	0.51
cse	16	91	7	4 (42%)	3.14	2.14 (31%)	0.79	0.53
mark1	15	36	6	4 (33%)	2.92	2.12 (27%)	0.73	0.53
keyb	19	170	8	5 (37%)	3.58	2.51 (29%)	0.72	0.50
ex1	20	138	9	5 (44%)	3.30	2.30 (30%)	0.66	0.46
dk512	15	30	6	4 (33%)	2.52	2.07 (17%)	0.63	0.52
s1	20	107	6	5 (16%)	2.71	2.14 (21%)	0.54	0.43
dk16	27	108	5	4 (20%)	2.44	2.20 (9%)	0.49	0.44
tbk*[a]	16	1024	2	2 (00%)	1.82	1.69 (7%)	0.45	0.42
tbk	32	1569	2	2 (00%)	1.81	1.75 (3%)	0.36	0.35

[a] A state-minimized version of the original circuit was used.

Table IV shows that the normalized average distance reduces to less than one in all cases. This indicates that the method is as effective in reducing the interstate distances as a method that adds edges by analyzing the STG topology. The entries of the table are sorted based on the normalized average distance in the original STG to highlight the fact that graphs with poorer connectivity (higher value) show greater reduction in the distances.

B. ATPG Performance on Circuits Synthesized with and Without Testability Enhancements

1) Circuit Synthesis: Four different circuits were synthesized for each FSM as follows:

a) using *jedi* (distributed with *sis*) with algorithmic option *o*;
b) using split-coding without clock control;
c) using split-coding with clock control using multiplexers;
d) using split-coding (with clock control) and the additional logic for observability enhancement.

In all four cases, synthesis was performed using the *rugged* (area optimization) script. The first circuit, therefore, would be the one obtained by a designer using the synthesis system in the standard way.

The split-code assignment for the synthesis of the second circuit of each FSM is described in the beginning of this section. The states were encoded in binary by randomly assigning codes to the α's and β's. No attempt was made to optimize these codes in order to obtain a minimum area of the synthesized circuits. Although this second circuit of each FSM was synthesized with the split-code assignment, no multiplexers were added for clock control. Consequently, the functionality of the α-steps of the split-code was not utilized. The purpose of presenting this set of results is to provide a baseline for the FSM's when the α-steps are utilized.

The third circuit for each FSM was synthesized using the same split-code assignment as in the second circuit. However, in the third circuit the α-steps are utilized by inserting a multiplexer to control the β-bits.

The fourth circuit for each FSM was synthesized to include observability enhancement in addition to split-coding (with clock control) as described in Section V. Since the added primary outputs, P_1 and P_2, are dependent on the state code, it was necessary to assign codes to the states before specifying the new outputs. Using the same state assignment as the second and third circuits, the specifications of P_1 and P_2 were added to the FSM specifications. The circuit was synthesized from the enhanced specification which led to the absorption of the added logic into the circuit.

2) Test Generation: For each FSM, test sequences were generated using *atpg* and *short_tests* (packages available with *sis*) for the four synthesized circuits. In each case, complete tests (100% fault efficiency) were generated.

Table V gives the performance of *short_tests* on the four circuits synthesized for each FSM. These circuits are identified in the table header by *jedi*, *split-code wo/clk-cntrl*, *split-code w/clk-cntrl*, and *split-code + obs*. There are three column entries for each circuit, providing information about the total number of faults (*#Flts*), the length of the test sequence (*Len*), and the total time (*Time*) required for test generation (with 100% efficiency). The entries in the table, as in Table IV, are ordered by the normalized average distance in decreasing order. Table VI provides analogous data for the performance of *atpg* on the synthesized circuits.

We will focus on test length as the measure of performance for split-coding in order to show improved controllability and

TABLE V
SYNTHESIS AND *short_tests* TEST-GENERATION RESULTS

Circuit	*jedi*			split-code wo/clk-cntrl			split-code w/clk-cntrl			split-code + obs		
	#Flts	Len	Time	#Flts	Len	Time	#Flts	Len	Time	#Flts	Len	Time
s510	482	324	17	545	417	294	569	259	93	688	268	47
planet	808	368	47	714	327	36	733	272	26	780	238	30
kirkman	279	206	5	318	380	7	329	163	4	368	158	5
s1494	861	669	42	824	581	38	845	305	30	892	314	31
s1488	850	691	41	799	617	45	821	325	31	853	353	34
sand	778	310	41	885	332	38	904	328	41	1110	364	54
s420	133	236	5	149	256	5	166	115	2	171	80	2
s208	176	222	5	133	213	4	153	100	3	162	96	2
s298[a]	2932	4151	26607	6592	3625	44698	6621	2357	26001	6651	2369	21561
tma	218	76	2	242	77	2	260	78	3	295	78	3
styr	642	372	24	759	394	39	774	315	37	788	305	36
s832	461	301	15	637	427	37	656	315	26	741	279	28
s820	488	341	26	577	441	28	596	360	27	645	281	20
pma	310	143	4	399	151	7	412	123	6	405	121	6
cse	297	148	5	325	181	5	335	125	4	345	123	5
mark1	151	70	1	182	62	1	192	70	2	205	74	2
keyb	320	231	7	328	206	5	344	219	8	360	182	9
ex1	383	109	5	408	104	6	425	95	6	496	102	7
dk512	110	69	1	142	55	1	155	55	1	159	47	1
s1	259	129	6	432	168	12	447	126	9	463	136	10
dk16	397	177	7	417	157	6	434	157	7	483	149	8
tbk*	293	165	5	365	200	9	381	217	7	391	186	7
tbk	350	196	8	903	477	67	919	482	67	1039	397	70

[a] The large test generation times for s298 are due to unoptimized synthesis.

TABLE VI
SYNTHESIS AND *atpg* TEST-GENERATION RESULTS

Circuit	*jedi*			split-code wo/clk-cntrl			split-code w/clk-cntrl			split-code + obs		
	#Flts	Len	Time	#Flts	Len	Time	#Flts	Len	Time	#Flts	Len	Time
s510	482	847	23	545	963	218	569	428	132	688	428	43
planet	808	956	49	714	795	30	733	424	23	780	392	26
kirkman	279	286	4	318	384	5	329	185	4	368	225	4
s1494	861	943	39	824	848	47	845	444	33	892	386	30
s1488	850	895	45	799	875	53	821	462	29	853	407	28
sand	778	428	33	885	410	30	904	412	26	1110	382	29
s420	133	256	3	149	318	3	166	153	2	171	139	2
s208	176	313	3	133	277	2	153	165	2	162	137	2
s298[a]	2932	4773	2799	6592	7828	13968	6621	4453	11189	6651	4119	9088
tma	218	102	2	242	98	2	260	106	3	295	106	4
styr	642	516	20	759	611	31	774	453	26	788	413	28
s832	461	431	10	637	574	18	656	434	16	741	437	18
s820	488	451	14	577	709	18	596	478	14	645	391	14
pma	310	168	5	399	226	7	412	189	7	405	154	6
cse	297	215	5	325	257	5	335	199	5	345	162	5
mark1	151	64	1	182	81	1	192	122	2	205	110	2
keyb	320	271	7	328	256	6	344	278	7	360	247	6
ex1	383	159	6	408	158	7	425	157	6	496	171	8
dk512	110	79	1	142	63	1	155	87	1	159	51	1
s1	259	191	6	432	210	10	447	209	10	463	236	10
dk16	397	260	9	417	216	7	434	232	8	483	219	9
tbk*	293	201	5	365	248	8	381	250	7	391	222	6
tbk	350	251	7	903	619	49	919	653	52	1039	524	45

[a] The large test generation times for s298 are due to unoptimized synthesis.

observability. Of the 22 circuits, 16 circuits in the *short_tests* experiment and 14 circuits in the *atpg* experiment show improvement over the *jedi* results when the circuits are synthesized with a split-code (with clock control) but without observability enhancement. The numbers marginally increase to 17 and 15 for the circuits synthesized with observability enhancement.

A comparison of the circuits synthesized without and with clock control shows that 15 circuits in both the *short_tests* and *atpg* experiments improved when the functionality of the α-steps is utilized. Likewise, the numbers increase to 17 and 19 for the two experiments when synthesis with observability enhancement is compared to the circuits synthesized without clock control.

The reduction in the test lengths is significantly high, as expected, for the FSM's which had poor connectivity in the original machine, i.e., high normalized average distance (top half of the table). The average reduction in the test length for

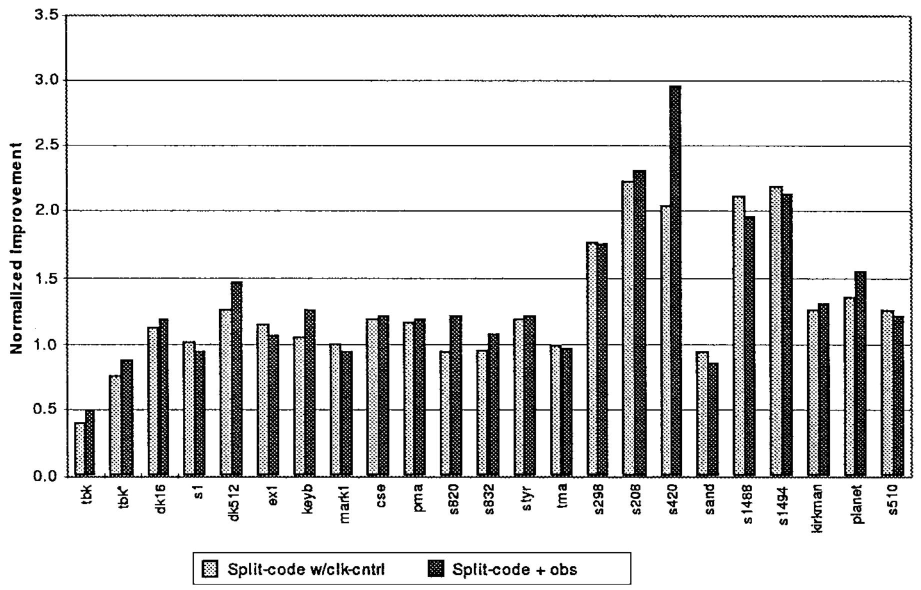


Fig. 7. *short_tests* test generation results—Comparison of test lengths.

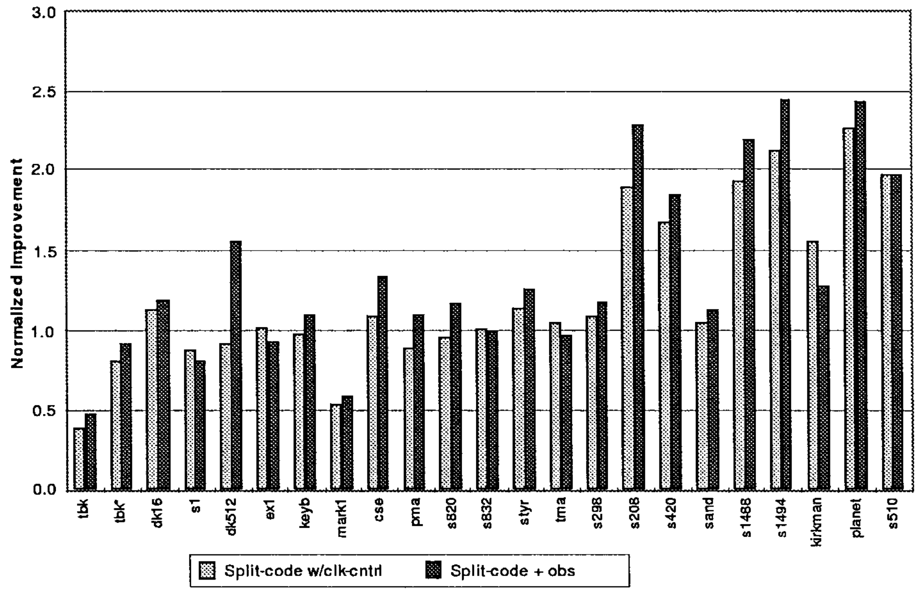


Fig. 8. *atpg* test generation results—Comparison of test lengths.

the top 11 circuits which were synthesized with clock control but without observability enhancement is 39.46% (*short_tests*) and 25.57% (*atpg*). With observability enhancement, these numbers are 39.39% (*short_tests*) and 30.89% (*atpg*), showing significant further improvement in *atpg* generated tests. Likewise, in comparison to the circuits synthesized without clock control, the average reduction in the test length for the top 11 circuits which were synthesized with clock control but without observability enhancement is 36.04% (*short_tests*) and 42.68% (*atpg*). With observability enhancement, these numbers are 35.96% (*short_tests*) and 46.79% (*atpg*).

To help visualize the magnitude of the improvement in the test length, the *normalized improvement* is displayed for the circuits in Figs. 7 and 8. The charts in each figure are normalized with respect to the first circuit, i.e., the *jedi*-synthesized circuit. The light and dark bars correspond to the *split-code w/clk-cntrl* and the *split-code* + *observability* circuits, respectively. Bars that extend above the 1.00 line on the charts indicate an improvement over the *jedi*-synthesized circuit. It is clear that the improvement is significantly high for the circuits in which the corresponding original circuit exhibits poor connectivity. The improvement is marginally higher for the circuits with enhanced observability.

A careful analysis of the data indicates that the circuits which showed no significant improvement in split-code-based synthesis had a large increase in the area of synthesis (see Table VII) as well as in the number of faults. This leads to our belief that the gains due to split-codes were offset by the larger fault set.

There are two ways to reduce the number of faults and the area cost in split-code-based circuits. In this experiment the binary encoding of the α and β fields was done randomly.

TABLE VII
AREA OF SYNTHESIZED CIRCUITS

Circuit	Area			
	jedi	split-code wo/clk-cntrl	split-code w/clk-cntrl	split-code + obs
s510	346	388	406	475
planet	593	499	519	555
kirkman	198	224	233	250
s1494	624	586	604	633
s1488	624	569	588	605
sand	543	635	649	767
s420	103	113	127	127
s208	132	107	121	117
s298[a]	2331	5144	5166	5184
tma	162	171	185	219
styr	453	531	552	571
s832	302	433	447	503
s820	328	395	409	437
pma	231	274	289	290
cse	212	224	233	249
mark1	101	116	130	134
keyb	237	231	244	250
ex1	255	283	297	342
dk512	84	105	114	116
s1	193	323	337	325
dk16	305	305	318	352
tbk*	206	256	265	271
tbk	238	647	662	746

[a] The size of S298 prevented area optimization by the synthesis program.

A tool to select these encodings can be developed which optimizes the synthesis area. Another approach would be to use tri-state buffers as shown in Fig. 5(a). This would add no logic overhead and only a minor overhead for routing control line C.

VII. Conclusion

As the practice of design moves toward high-level synthesis, it becomes feasible to consider schemes aimed at enhancing testability through synthesis. One such scheme is proposed here as a systematic way to improve the testability of FSM's by a novel scheme for symbolic state-encoding and clock control. We show, both theoretically and through experimental evidence, that the resulting gains in testability could lead to substantially shorter test sets with the same fault efficiency. The full benefits of split-coding can be realized by integrating the knowledge about enhanced navigability between states in the ATPG tool.

Appendix
Proofs

Here, we present the proofs of several claims made in the paper.

Observation 2: A modulo-N $(N = 2^n)$ counter is synthesized using two clocks (as described in Section II) with the ith state in the cycle, S_i, assigned to the binary code of integer i. If FF's are partitioned linearly, then the average distance between the states in the original machine and the test machine are approximately $N/2$ and $3N/8$, respectively.

Proof: Suppose the left-most $n-p$ bits are β-bits. If $a_{n-1}\cdots a_1a_0 \rightarrow b_{n-1}\cdots b_1b_0$ is a transition with $C = 0$, then $a_{n-1}\cdots a_1a_0 \rightarrow a_{n-1}\cdots a_pb_{p-1}\cdots b_0$ is the corresponding α-transition.

The average distance between any pair of states in the normal machine is $\Sigma_{i=0}^{N-1} i/N = (N-1)/2$.

The α-transitions, created by $C = 1$, coincide with the corresponding $\alpha\beta$-transitions except for the states encoded $a_{n-1}\cdots a_p1\cdots 1$. In these cases the new transitions are $a_{n-1}\cdots a_p1\cdots 1 \rightarrow a_{n-1}\cdots a_p0\cdots 0$. With these additional edges, the STG of the circuit transforms to a cycle of super-nodes where each super-node is itself a cycle of length 2^p.

The average distance between two states in the same sub-cycle is $2^{p-1} - 0.5$. The average distance between the two states separated by k sub-cycles is $(2^{p-1} - 0.5) + (2^{p-1} - 0.5) + k\cdot(2^p) + 1 = (k+1)\cdot 2^p$, where $0 \le k \le 2^{n-p} - 2$. Thus, the average distance between any pair of states is $2^{n-1} - 2^{p-1} + 2^{2p-n-1} - 2^{p-1-n}$. The minimum average distance is approximately $3N/8$ when $p = n-1$. ●

A. Split-Codes and Their Properties

Here, we present various properties of split-codes and their proofs leading to the proof of Theorem 1. For completeness and rigor we restate the definition of a split-code.

Consider the mapping M: $\mathcal{N}^+ \rightarrow [m] \times [2^k]$, where $[x]$ denotes the set $\{0, 1, \ldots, x-1\}$ and k, m are some positive integers with $k \le m$, defined inductively as follows:

i) $M(0) = \langle 0, 0\rangle$

ii) $M(j+1) = \langle \alpha+1, \beta+2^\alpha\rangle\ \forall j \ge 0,$

$$\text{where } M(j) = \langle \alpha, \beta\rangle. \tag{1}$$

Here and subsequently, arithmetic operations on the elements of $[m]$ and $[2^k]$ are understood to be *modulo-*m and *modulo-*2^k, respectively.

Observation 3: If $M(j) = \langle\alpha,\beta\rangle$, then $M(j+p\cdot m) = \langle \alpha, \beta - p\rangle$.

Proof: From the definition of function M

$$M(j+m) = \langle \alpha, \beta + 2^\alpha \cdots + 2^{\alpha+m-1}\rangle.$$

Since the exponents are modulo-m numbers, upon rearranging them we obtain

$$\begin{aligned} M(j+m) &= \langle \alpha, \beta + 2^0 \cdots + 2^{m-1}\rangle = \langle \alpha, \beta + 2^m - 1\rangle \\ &= \langle \alpha, \beta - 1\rangle. \end{aligned}$$

The last equation is due to the fact that 2^k divides 2^m as $k \le m$. Thus, by the repeated application of this result

$$\begin{aligned} M(j+p\cdot m) &= M(j+m\cdots+m) = \langle \alpha, \beta - 1 \cdots - 1\rangle \\ &= \langle \alpha, \beta - p\rangle \quad \bullet \end{aligned}$$

Observation 4: i) $M(m\cdot 2^k) = M(0)$, and ii) M: $[m\cdot 2^k] \rightarrow [m] \times [2^k]$ is a one-to-one correspondence.

Proof: i) From Observation 3

$$M(m\cdot 2^k) = M(0 + m\cdot 2^k) = \langle 0, 0 - 2^k\rangle = \langle 0, 0\rangle = M(0).$$

ii) Assume that there are indexes $0 \le j \le j+i < m\cdot 2^k$ such that $M(j) = M(j+i)$. Let $M(j) = \langle\alpha,\beta\rangle$. From the definition of function M, $\alpha = \alpha + i$. So i must be divisible by m, i.e., $i = p\cdot m$ for some integer p. From the assumption, $0 \le p < 2^k$. From Observation 3, $M(j+i) = M(j+p\cdot m) = \langle\alpha, \beta - p\rangle$.

Since $\beta = \beta - p, p$ should be divisible by 2^k. But since p is strictly less than 2^k, $p = 0$. Thus, $i = 0$ or $j = j + i$. ●

Once again for convenience, we will assume that arithmetic operations on the indexes of M are modulo-$m \cdot 2^k$.

Definition: For any positive integers k, m with $k \le m$, the bijection M: $[m\cdot 2^k] \rightarrow [m]\times[2^k]$ given by (1) (alternatively, the range of the function M) is called a *split-code* with parameters k, m.

Definition: If $\langle\alpha,\beta\rangle = M(i)$ and $\langle\gamma,\delta\rangle = M(j)$, then their *difference*, $j - i$, is denoted by $\Delta(\langle\gamma,\delta\rangle, \langle\alpha, \beta\rangle)$.

Example: For a split-code of $m = 3$, $k = 2, \langle 2,3\rangle = M(2)$ and $\langle 0,2\rangle = M(6)$, so $\Delta(\langle 0,2\rangle, \langle 2,3\rangle) = 6 - 2 = 4$, and $\Delta(\langle 2,3\rangle, \langle 0,2\rangle) = 2 - 6 = -4 = 8$.

Observation 5: i) $\Delta(\langle\alpha,\beta'\rangle, \langle\alpha,\beta\rangle) = m\cdot(\beta - \beta')$.

ii) $\Delta(\langle\alpha+1,\beta\rangle, \langle\alpha,\beta\rangle) = 1 + m\cdot 2^\alpha$. So, in particular, for $\alpha \ge k$ the difference will be one.

Proof: i) Suppose $\langle\alpha,\beta\rangle = M(j)$. Due to the cyclic nature of the first component, $\langle\alpha,\beta'\rangle = M(j + p\cdot m)$ for some p. From Observation 3, $M(j + p\cdot m) = \langle\alpha, \beta - p\rangle$ giving $p = \beta - \beta'$. Thus, $\Delta(\langle\alpha,\beta'\rangle, \langle\alpha,\beta\rangle) = \Delta(M(j + p\cdot m), M(j)) = p\cdot m = m\cdot(\beta - \beta')$.

ii) Let $\langle\alpha,\beta\rangle = M(j)$ and $\langle\alpha+1,\beta\rangle = M(j+i)$. Thus, $M(j+i-1) = \langle\alpha,\beta'\rangle$ where $\beta' = \beta - 2^\alpha$. From the first part of the proof, $i - 1 = \Delta(M(j+i-1), M(j)) = \Delta(\langle\alpha,\beta'\rangle, \langle\alpha,\beta\rangle) = m\cdot(\beta-\beta')$. So $i = m\cdot 2^\alpha + 1$. ●

Definition: A sequence $\langle\alpha_0,\beta_0\rangle, \langle\alpha_1,\beta_1\rangle, \ldots, \langle\alpha_p,\beta_p\rangle$ is *monotonic* if $\Sigma_{i=0}^{p-1}\Delta(\langle\alpha_{i+1},\beta_{i+1}\rangle, \langle\alpha_i,\beta_i\rangle) \le m\cdot 2^k$, where the summation is **not** *modulo* $m\cdot 2^k$.

Example: In the split-code shown in Table I, the sequence $M(1), M(5), M(7), M(10)$ is monotonic but the sequence $M(1), M(7), M(5)$ is not monotonic.

Lemma 1: All unit-step sequences of the form $\langle\alpha,\beta_0\rangle, \langle\alpha+1,\beta_1\rangle, \ldots, \langle\alpha+m-1,\beta_{m-1}\rangle, \langle\alpha,\beta_m\rangle$ are monotonic.

Proof: If step $(\langle\alpha+i,\beta_i\rangle, \langle\alpha+i+1,\beta_{i+1}\rangle)$ is an $\alpha\beta$-step (i.e., adjacent codes), then from the definition $\Delta(\langle\alpha+i+1,\beta_{i+1}\rangle, \langle\alpha+i,\beta_i\rangle) = 1$.

On the other hand, if $(\langle\alpha+i,\beta_i\rangle, \langle\alpha+i+1,\beta_{i+1}\rangle)$ is an α-step, then $\beta_{i+1} = \beta_i$. From Observation 5, $\Delta(\langle\alpha+i+1,\beta_{i+1}\rangle, \langle\alpha+i,\beta_i\rangle) = m\cdot 2^{\alpha+i} + 1$. In case $\alpha + i \ge k$, the distance reduces to one since $m\cdot 2^{\alpha+i}$ is divisible by $m\cdot 2^k$.

Thus

$$\begin{aligned}\sum_{i=0}^{m-1} &\Delta(\langle\alpha+i+1,\beta_{i+1}\rangle, \langle\alpha+i,\beta_i\rangle)\\ &\le \sum_{\alpha+i=0}^{k-1}(m\cdot 2^i + 1) + \sum_{\alpha=+i=k}^{m-1} 1\\ &= \left(\sum_{j=0}^{k-1} m\cdot 2^j\right) + k + (m-k)\\ &= m\cdot(2^k - 1) + m = m\cdot 2^k.\end{aligned}$$

Note that the inequality will be strict if $\langle\alpha,\beta_0\rangle$ and $\langle\alpha,\beta_m\rangle$ are different. ●

Corollary: For any two codes $\langle\alpha,\beta'\rangle$ and $\langle\alpha,\beta''\rangle$ that differ only in the second component, there exists a monotonic unit-step sequence $\langle\alpha,\beta_0 = \beta'\rangle, \langle\alpha+1, \beta_1\rangle, \ldots, \langle\alpha+m-1, \beta_{m-1}\rangle, \langle\alpha,\beta_m = \beta''\rangle$.

Proof: Construct the sequence $\langle\alpha,\beta'\rangle = \langle\alpha,\beta_0\rangle, \langle\alpha+1, \beta_1\rangle, \ldots, \langle\alpha+m-1, \beta_{m-1}\rangle, \langle\alpha,\beta_m\rangle$ with the following stipulation: for all i, $\beta_{i+1} = \beta_i$ if the $(\alpha+i)$th bit of $\beta'' - \beta'$ (mod 2^k) is zero, otherwise, $\beta_{i+1} = \beta_i + 2^{\alpha+i}$. Here, the kth and the higher bits of $\beta'' - \beta'$ (mod 2^k) are zero.

From the construction, $\beta_m = \beta''$. From the lemma this sequence is monotonic. ●

Theorem 1: For any sequence of split-code pairs $S_{i,j}$: $M(i), M(i+1), \ldots, M(i+q)$ $(= M(j))$, there exists a unit-step sequence

$$M(i), M(i+n_1), M(i+n_1+n_2), \cdots,$$
$$M(i+n_1+n_2+\cdots+n_p)(= M(j))$$

with $p < 2m$. Further, this sequence is contained in $S_{i,j}$.

Proof: Let $M(i) = \langle\alpha_1, \beta_1\rangle$ and $M(j) = \langle\alpha_2,\beta_2\rangle$. The desired sequence can be constructed in two parts. The initial subsequence is $M(i), M(i+1), \ldots, M(i+p)$ where $M(i+p)$ is the first code with the first component α_2, say $M(i+p) = \langle\alpha_2,\beta'\rangle$ for some β'. So $p < m$.

The final subsequence, from $M(i+p) = \langle\alpha_2,\beta'\rangle$ to $M(j) = \langle\alpha_2,\beta_2\rangle$ with no more than m-steps is assured by Lemma 1. ●

ACKNOWLEDGMENT

The authors would like to thank Mr. F. Geng for his assistance with some of the experiments. They are also grateful to Dr. V. D. Agrawal for many helpful discussions.

REFERENCES

[1] V. D. Agrawal and K.-T. Cheng, "Finite state machine with embedded test function," *J. Electron. Testing: Theory and Applications*, vol. 1, pp. 221–228, Oct. 1990.
[2] V. D. Agrawal, S. C. Seth, and J. S. Design, "Design for testability and test generation with two clocks," in *Proc. 4th Int. Symp. VLSI Design*, Jan. 1991, pp. 112–117.
[3] S. Baeg and W. A. Rogers, "Hybrid design for testability combining scan and clock line control and method for test generation," in *Proc. Int. Test Conf.*, 1994, pp. 340–349.
[4] K. T. Cheng and V. D. Agrawal, "Design of sequential machines for efficient test generation," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, 1989, pp. 358–361.
[5] ——, "Methods for synthesizing testable sequential circuits," *AT&T Technical Journal*, pp. 64–86, Jan. 1991.
[6] ——, "State assignment for testable design," in *Proc. Int. J. Computer-Aided Design*, vol. 3, pp. 291–297, 1991.
[7] K. L. Einspahr, S. C. Seth, and V. D. Agrawal, "Clock partitioning for testability," in *Proc. 3rd IEEE Great Lakes Symp. VLSI Design*, Mar. 1993, pp. 42–46.
[8] ——, "Improving circuit testability by clock control," in *Proc. 6th IEEE Great Lakes Symp. VLSI Design*, Mar. 1996, pp. 288–293.
[9] W.-C. Fang and S. K. Gupta, "Clock grouping: A low cost DFT methodology for delay testing," in *Proc. Design Automation Conf.*, 1994, pp. 94–99.
[10] H. Fujiwara, Y. Nagao, T. Sasao, and K. Kinoshita, "Easily testable sequential machines with extra inputs," *IEEE Trans. Comput.*, vol. C-24, pp. 821–826, Aug. 1975.
[11] F. H. Hsu and J. H. Patel, "Design for testability using state distances," *J. Electron. Testing: Theory and Applications (JETTA)*, vol. 11, no. 1, pp. 93–100, Aug. 1997.
[12] S. Kanjilal, S. T. Chakradhar, and V. D. Agrawal, "Test function embedding algorithms with applications to interconnected finite state ma-

chines," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 1115–1117, Sept. 1995.

[13] J. Lee and J. H. Patel, "ARTEST: An architecture level test generator for data path faults and control faults," in *Proc. Int. Test Conf.*, Oct. 1991, pp. 729–738.

[14] S. K. Mehta, S. C. Seth, and K. L. Einspahr, "Synthesis for testability by two-clock control," in *Proc. 10th Int. Conf. VLSI Design (VLSI Design '97)*, pp. 279–283.

[15] S. Ohtake, T. Masuzawa, and H. Fujiwara, "A nonscan DFT method for controllers to achieve complete fault efficiency," in *Proc. Asian Test Symp.*, 1998, pp. 204–211.

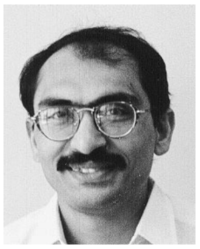
[16] R. G. R. Gupta and M. A. Breuer, "BALLAST: A methodology for partial scan design," in *Proc. 19th Int. Symp. Fault-Tolerant Computing (FTCS-19)*, June 1989, pp. 118–125.

[17] K. B. Rajan, D. E. Long, and M. Abramovci, "Increasing testability by clock transformation (getting rid of those darn states)," in *Proc. 14th IEEE VLSI Test Symp.*, Apr./May 1996, pp. 224–230.

[18] T. Villa and A. Sangiovanni-Vincentelli, "Nova: State assignment of finite state machines for optimal two-level logic implementation," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 905–924, Sept. 1990.

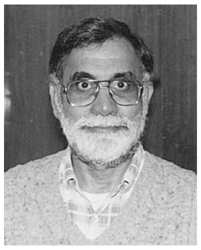
Kent L. Einspahr received the M.S. and Ph.D. degrees from the University of Nebraska-Lincoln in 1985 and 1991, respectively.

He is currently a Professor of Computer Science at Concordia University, Seward, NE, and the Chair of the Mathematics and Computer Science Department. His current research interests are in test generation, design for testability, synthesis for testability, and design verification and diagnosis. His other interests include parallel processing, especially in its application to computer-aided design, and pattern recognition.



Shashank K. Mehta received the MSc (Physics) and MTech degrees from Indian Institute of Technology, Kanpur, India, and the Ph.D. degree from University of Nebraska-Lincoln.

Since 1990, he has been with University of Pune, Pune, India, where he is currently a Reader in Computer Science Department. His research interests include VLSI test generation, evaluation of test performance, and estimation of chip quality level. He is also interested in geometric algorithms, solid modeling, and category theory.



Sharad C. Seth received the B.E. (Hons.) degree from University of Jabalpur, Jabalpur, India, the M.Tech. degree from IIT Kanpur, Kanpur, India, and the Ph.D. degree from University of Illinois, Urbana.

Since 1970, he has been with the University of Nebraska-Lincoln where currently he is a Professor of Computer Science and Engineering and the Director of the Center for Communication and Information Science. His current research interests in the VLSI testing area include developing an integrated approach to design-verification and manufacturing tests, device-tester data analysis for estimating quality level, and synthesis for testability. His other research interests include document image analysis and geographical information systems.