On Test Set Preservation of Retimed Circuits

Aiman El-Maleh[1], Thomas Marchok[2], Janusz Rajski[3], Wojciech Maly[2]

[1] MACS Laboratory, McGill University, 3480 University St., Montreal, Canada, H3A 2A7

[2] ECE Department, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213

[3] Mentor Graphics Corp., 8005 SW Boeckman Rd., Wilsonville, OR 97070-7777

Abstract— Recently, it has been shown that retiming has a very strong impact on the run time of sequential, structural automatic test pattern generators (ATPGs), as well as the levels of fault coverage and fault efficiency attained. In this paper, we show that retiming preserves testability with respect to a single stuck-at fault test set by adding a prefix sequence of a pre-determined number of arbitrary input vectors. Experimental results show that high fault coverages can be achieved on high performance circuits optimized by retiming with a much less CPU time (a reduction of two orders of magnitude in several instances) than if ATPG is attempted directly on those circuits.

I. Introduction

Due to the steadily increasing demand for high performance integrated circuits, performance optimization must be considered as a key step during each stage of the synthesis process. Synthesis tools must be able to synthesize circuits that meet performance constraints while optimizing other design parameters like area, power, and testability.

One of the effective techniques for performance optimization of sequential circuits is retiming. Retiming ([4], [5]) involves moving registers across portions of combinational logic in order to minimize the clock cycle time or the number of registers used. It has been determined ([6], [8]) that retiming sequential circuits for performance optimization increases the run time required by sequential, structural automatic test pattern generators (ATPGs), as well as decreases the levels of fault coverage and fault efficiency attained. In several instances, the increase in execution time is more than two orders of magnitude.

In order to reliably predict the behavior of, or derive tests for, a sequential circuit that does not have a global reset state, a synchronizing sequence is applied to drive it to a known state. In this paper, we study the preservation of synchronizing sequences and test sets under the retiming transformation. We show that a synchronizing sequence for a circuit can be mapped to synchronize any of its corresponding retimed circuits to an equivalent state. More importantly, we show that a test set generated for a circuit is preserved on its corresponding retimed circuit by adding a prefix sequence of a pre-determined number of arbitrary input vectors. The implications of this result on the testability of retimed circuits and the cost of test pattern generation are then discussed.

This work has been supported by research grants from the Semiconductor Research Corp under contract no. 94-DC-068, Intel, the "Natural Sciences and Engineering Research Council of Canada", and a scholarship from the Quebec "Fonds pour la Formation de Chercheurs et l'Aide à la Recherche".

The rest of the paper is organized as follows. Section II introduces the basic concepts and definitions used. Section III presents an overview of retiming. Section IV presents the effect of retiming on synchronizing sequences and test sets. Section V demonstrates testability preservation of retiming on a set of circuits synthesized from MCNC finite-state machine descriptions. Section VI summarizes the main results of this paper.

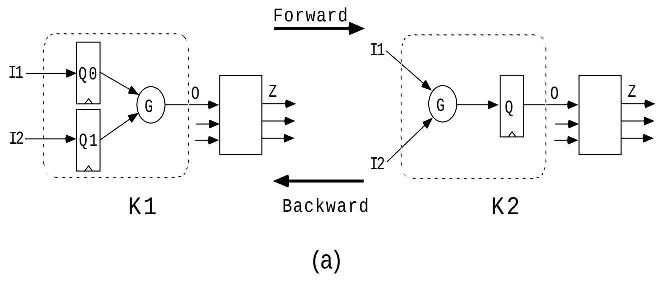
II. Basic Concepts and Definitions

In our work, we focus on synchronous sequential circuits. We assume that a synchronous sequential circuit is composed of combinational logic gates and edge-triggered D flip-flops (DFFs). For circuits that do not have a global reset state, initialization is achieved by synchronizing sequences. A *synchronizing sequence* for a machine K is an input sequence that brings the machine to a known and unique state (or a set of equivalent states [11]), that can be determined without any knowledge of the output response or the initial state of the machine [3]. The final state reached by the synchronizing sequence is called a *reset state*. A state that can be reached from the reset state via some input sequence is called a *valid state*. A state which cannot be reached from the reset state is called an *invalid state*.

The following definitions on state equivalence and state distinguishability are taken from [3]. Two states q and q' in a machine K are said to be *equivalent* if and only if the input/output (I/O) behavior of K starting in initial state q is the same as that of K starting in initial state q'. Two states q and q' in a machine K are said to be *distinguishable* if and only if there exists a finite input sequence that yields one output sequence when K is started in state q, and a different output sequence when K is started in state q'. Equivalence and distinguishability of two states in two different machines can be defined similarly.

Given two machines K and K', if for every state in machine K' there exists at least one equivalent state in machine K, then machine K is said to *space-contain* machine K', denoted $K \supseteq_s K'$. If for every state in K there is at least one equivalent state in K' and vice versa i.e., $K \subseteq_s K'$ and $K \supseteq_s K'$, then K and K' are said to be *space-equivalent*, denoted $K \equiv_s K'$.

Given a machine K, let K_i denote the set of states reachable from any state in K after applying i transitions. It should be observed that $K_0 \equiv_s K$ and that $K_i \supseteq_s K_{i+1}$. A machine K is said to *time-contain* machine K', denoted $K \supseteq_t K'$, if there exists a time i such that $K \supseteq_s K'_i$. If



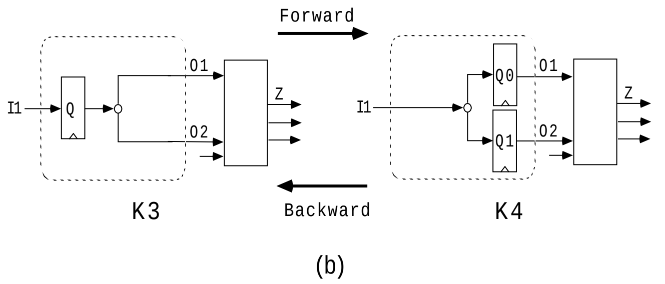


Fig. 1. (a) Retiming forward/backward across a single-output combinational gate, (b) Retiming forward/backward across a fanout stem.

after applying N transitions to machine K' every state in K' has at least one equivalent state in K i.e., $K \supseteq_s K'_N$, then K is said to *N-time-contain* K', denoted $K \supseteq_{Nt} K'$. If $K \supseteq_{N1t} K'$ and $K' \supseteq_{N2t} K$, then machine K is said to be *N-time-equivalent* to machine K', denoted $K \equiv_{Nt} K'$, where $N = max(N1, N2)$. It should be observed that $K \equiv_{Nt} K'$ is equivalent to $K_N \equiv_s K'_N$. Space-containment and space-equivalence are stronger relations and imply time-containment and time-equivalence, respectively.

In order to detect a single stuck-at fault in a sequential circuit, the machine has to be placed in a state which excites the fault, and then the fault effect must be propagated to a primary output. If no explicit reset state is assumed, then memory elements are assumed to start from the unknown initial state.

To process the unknown initial state in addition to "0" and "1", simulation algorithms use an extra logic value, denoted by x. There is a loss of information associated with the use of 3-valued logic, which may lead to pessimistic results i.e., a synchronizing sequence for a circuit might not be considered a valid synchronizing sequence using 3-valued logic simulation. In our work, we call a synchronizing sequence (or a test) derived based on 3-valued simulation *structural-based*, otherwise it is called *functional-based.* It should be observed that functional-based synchronizing sequences (or tests) correspond to those derived based on the state transition graph of a circuit.

Let K be a single stuck-at fault testable circuit. Let K' be a circuit resulting from applying a transformation T to circuit K. The transformation T is said to be single stuck-at fault *testability preserving* if K' is guaranteed to be single stuck-at fault testable. Transformation T is said to be single stuck-at fault *test-set preserving* if any complete single stuck-at fault test set for K is also a complete single stuck-at fault test set for K'.

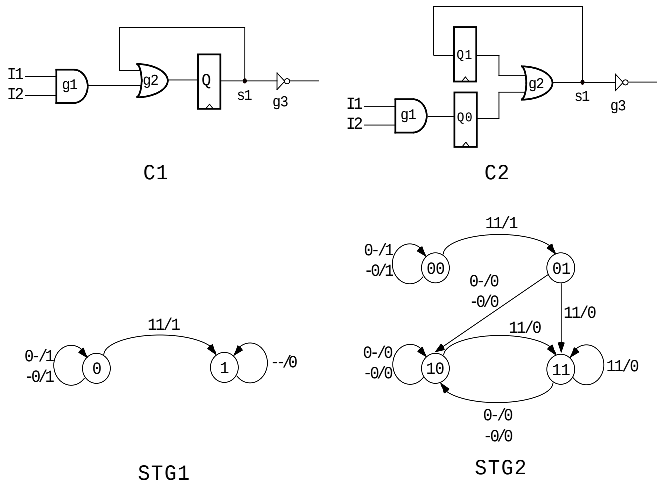


Fig. 2. An example of a backward retiming move across a single-output combinational gate.

III. Retiming: An Overview

Retiming can be thought of as a collection of atomic transformations which move sequential elements forward or backward across a single-output combinational gate or a fanout stem, as shown in Fig. 1. In this figure, $I1, I2$ and Z are assumed to be functions of the primary inputs and the state variables. Z is assumed to be a primary output, G is a combinational gate, and $Q, Q0$, and $Q1$ are sequential elements (DFFs).

For the purpose of retiming, a sequential circuit is modeled as a finite edge-weighted directed graph $G = (V, E, W)$. The vertices V represent either an input/output pin or a combinational logic block. For simplicity and without loss of generality, we assume that vertices represent single-output combinational gates and fanout stems. The edges E correspond to interconnections between input pins, combinational logic gates, fanout stems, and output pins. The weights W assign to each edge a non-negative integer representing the number of sequential elements along the interconnection between the two vertices. Retiming computes an edge weight that minimizes a cost function (e.g., the length of the longest combinational path in the circuit, the number of memory elements in the circuit, or a combination of both).

Fig. 2 shows an example of a circuit that is retimed for increased performance. Assuming that the delay of a combinational gate is related to the number of its inputs, the circuit $C1$ has a clock period of four delay units. The performance of the circuit $C1$ can be improved by one delay unit through retiming resulting in the circuit $C2$. As can be seen from the figure, while the state transition graph (STG) of $C1$, has no equivalent states, the STG of $C2$ has three equivalent states, namely $\{01, 10, 11\}$. Thus, as was illustrated in [2], retiming can create and add equivalent states. Furthermore, it should be observed that for every state in $C2$ there is an equivalent state in $C1$ and vice versa. State $\{00\}$ in $C2$ is equivalent to state $\{0\}$ in $C1$, and states $\{01, 10, 11\}$ in $C2$ are equivalent to state $\{1\}$ in $C1$. Thus, this implies that $C1$ and $C2$ are space-equivalent i.e., $C1 \equiv_s C2$. This observation leads us to the following result.

Lemma 1: Let K' be a circuit resulting from a retiming of

K using forward and backward retiming moves across single-output combinational gates. Then, $K \equiv_s K'$.

Due to space limitations, the reader is referred to [1] for proofs of all theorems and lemmas presented in this paper.

Let K' be a circuit resulting from a retiming of K. It was observed in [4] that, in general, it is not true that for every state in K there is an equivalent state in K' i.e., $K \not\subseteq_s K'$. This is due to the fact that when K is retimed backward across a fanout stem node, there exist assignments on the DFFs in K that are inconsistent with the values produced by the logical structure of the circuit i.e., when different values are assigned on the branches of a fanout stem. Such assignments have no equivalent assignment on the DFFs in the circuit K'. Similarly, not every state in K' has an equivalent state in K (i.e., $K \not\supseteq_s K'$) when K is retimed forward across a fanout stem node. Thus, this implies that the two circuits K and K' may not be space-equivalent i.e., $K \not\equiv_s K'$.

Let F and B represent the maximum number of forward and backward retiming moves, respectively, across any node when K is retimed to K'. It was shown in [4] that for every state in K_B, there is an equivalent state in K' i.e., $K' \supseteq_s K_B$ or $K' \supseteq_{Bt} K$. Similarly, for every state in K'_F there is an equivalent state in K i.e., $K \supseteq_s K'_F$ or $K \supseteq_{Ft} K'$. This implies that K and K' are time-equivalent since there exists a time $N = max(F, B)$ such that $K_N \equiv_s K'_N$ or $K \equiv_{Nt} K'$. Based on Lemma 1, F and B can be tightened to represent the maximum number of forward and backward retiming moves across any fanout stem node, respectively.

Lemma 2: Let K' be a circuit resulting from a retiming of K. Let F and B represent the maximum number of forward and backward retiming moves across any **fanout stem** node in K, then:

1. $K' \supseteq_{Bt} K$,
2. $K \supseteq_{Ft} K'$,
3. $K \equiv_{Nt} K'$, where $N = max(F, B)$.

Lemma 2 is an important result as it clearly describes the relation between the state transition graphs of a circuit and any of its corresponding retimed circuits. Based on this lemma, we show in the following section how a synchronizing sequence or a test for a retimed circuit, can be derived based on a synchronizing sequence or a test for its corresponding original circuit.

IV. Testability Preservation Under Retiming

To detect a fault in a sequential circuit, assuming unknown initial state, both the fault-free and the faulty circuits need to be synchronized. Thus, it is important to study the effect of retiming on synchronizing sequences for the fault-free and the faulty circuits. In this section, we first derive the conditions under which synchronizing sequences are preserved under retiming. Then, we show that a test set for a circuit is preserved on its corresponding retimed circuit by adding a prefix sequence of a pre-determined number of arbitrary input vectors.

A. Preservation of Synchronizing Sequences for Fault-Free Circuits

In this subsection, we determine the conditions under which synchronizing sequences for fault-free circuits are preserved under retiming. We first show that retiming preserves structural-based synchronizing sequences.

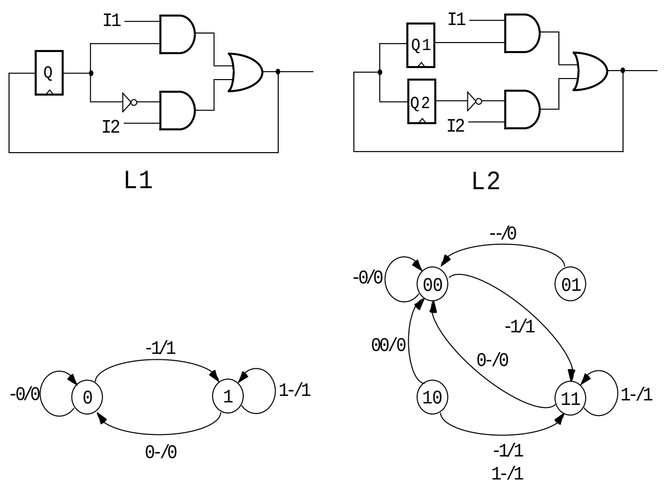


Fig. 3. An example of a forward retiming move across a fanout stem.

Theorem 1: Let K' be a circuit resulting from a retiming of K. If a structural-based synchronizing sequence I synchronizes K to a state q, then I synchronizes K' to a state q' equivalent to q.

Theorem 1 is an important result as it guarantees that a circuit and any of its corresponding retimed circuits are structurally synchronized to equivalent states and hence exhibit the same input/output behavior after synchronization.

For example, referring to Fig. 2, the input vector $\langle 11 \rangle$ synchronizes the circuit $C1$ to state $\{1\}$. Similarly, the input vector $\langle 11 \rangle$ synchronizes the circuit $C2$ to the equivalent states $\{01, 11\}$. It should be observed that state $\{1\}$ in $C1$ is equivalent to states $\{01, 11\}$ in $C2$. Thus, the input vector $\langle 11 \rangle$ synchronizes both circuits to equivalent states.

While structural-based synchronizing sequences are preserved under retiming, the same is not true for functional-based synchronizing sequences. This is due to the fact that the retimed circuit might contain states that have no equivalent state in the original circuit.

Observation 1: A functional−based synchronizing sequence for a circuit K does not necessarily synchronize any circuit resulting from a retiming of K.

Example 1: The circuit $L1$ in Fig. 3 is transformed to circuit $L2$ using a forward retiming move across a fanout stem node. The vector $\langle 11 \rangle$ is a functional-based (but not structural-based) synchronizing sequence for the circuit $L1$ and synchronizes $L1$ to the state $\{1\}$. However, the same vector does not synchronize the circuit $L2$.

We next derive the conditions under which functional-based synchronizing sequences are preserved on retimed circuits.

Lemma 3: Let K and K' be two circuits and $K \supseteq_s K'$. If a functional-based synchronizing sequence I synchronizes K to a state q, then the same sequence I synchronizes K' to a state q' equivalent to q.

Thus, from Lemma 2, it follows that every functional-based synchronizing sequence for a circuit synchronizes to an equivalent state any circuit resulting from a retiming of K using forward and/or backward retiming moves across single-

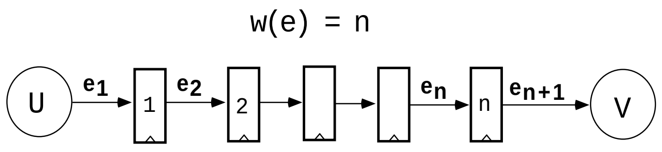


Fig. 4. An edge of weight n connecting vertex u to vertex v.

output combinational gates, and backward retiming moves across fanout stems.

Theorem 2: Let K' be a circuit resulting from a retiming of K, and P be a sequence of arbitrary input vectors of length equal to the maximum number of forward retiming moves across any fanout stem node in K. If a functional-based synchronizing sequence I synchronizes K to a state q, then the sequence $P \cup I$ synchronizes K' to a state q' equivalent to q.

Theorem 2 provides the mechanism for deriving synchronizing sequences for retimed circuits based on functional-based synchronizing sequences for their corresponding original circuits. For example, we have shown in Example 1 that the vector $\langle 11 \rangle$ is a functional-based synchronizing sequence for the circuit $L1$ (shown in Fig. 3), but not a synchronizing sequence for the circuit $L2$. It can be readily verified that each of the following sequences $\langle 00, 11 \rangle$, $\langle 01, 11 \rangle$, $\langle 10, 11 \rangle$, and $\langle 11, 11 \rangle$ is a functional-based synchronizing sequence for $L2$ and synchronizes it to state $\{11\}$ which is equivalent to state $\{1\}$ in $L1$.

B. Preservation of Synchronizing Sequences for Faulty Circuits

In this subsection, we study the effect of retiming on synchronizing sequences for faulty circuits. We use the term *corresponding faults* to relate faults in a circuit, K, and any circuit, K', resulting from a retiming of K. Let e be an edge from vertex u to vertex v in K. An edge e of weight n is considered to be divided into $n+1$ lines as shown in Fig. 4. A fault on a line e_i in K corresponds to all faults on the lines $e'_{i_1}, e'_{i_2}, \cdots, e'_{i_{m+1}}$ in K' created by placing m DFFs on line e_i. Similarly, a fault on a line e'_i in K' corresponds to all faults on the lines $e_{i_1}, e_{i_2}, \cdots, e_{i_{m+1}}$ in K that are merged to form the edge e'_i by removing the m DFFs separating them.

For example, the following constitute corresponding faults in the circuits in Fig. 1(a):

- Stuck-at-0 (stuck-at-1) fault at line $I1 - Q0$ in $K1$ and stuck-at-0 (stuck-at-1) fault at line $I1 - G$ in $K2$.
- Stuck-at-0 (stuck-at-1) fault at line $Q0 - G$ in $K1$ and stuck-at-0 (stuck-at-1) fault at line $I1 - G$ in $K2$.
- Stuck-at-0 (stuck-at-1) fault at line $G - Q$ in $K2$ and stuck-at-0 (stuck-at-1) fault at line $G - O$ in $K1$.
- Stuck-at-0 (stuck-at-1) fault at line $Q - O$ in $K2$ and stuck-at-0 (stuck-at-1) fault at line $G - O$ in $K1$.

It should be observed that faults outside the modified region in a retimed circuit are in one-to-one correspondence with faults outside the modified region in the original circuit. In general, for every fault on a line in a retimed circuit, there is at least one corresponding fault in the original circuit.

In the following, we show that for every fault in a retimed circuit, there exists a corresponding fault in the original circuit such that a synchronizing sequence for the original faulty circuit can be mapped to serve as a synchronizing sequence for the corresponding faulty retimed circuit. This result is stated first for each of the atomic retiming transformations.

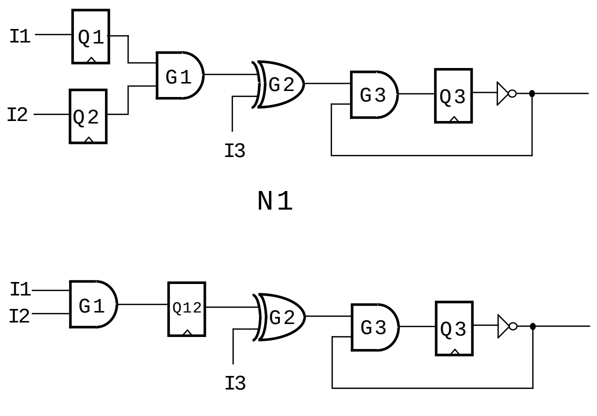


Fig. 5. An example of a forward retiming move across a single-output combinational gate.

Lemma 4: Let K' be a circuit resulting from a forward retiming move across a single-output combinational gate or a fanout stem in K. For every fault f' in K', there exists a corresponding fault f in K such that if a sequence I synchronizes K^f to a state q, then the sequence $p \cup I$, where p is an arbitrary input vector, synchronizes $K'^{f'}$ to a state q' equivalent to q.

Lemma 5: Let K' be a circuit resulting from a backward retiming move across a single-output combinational gate or fanout stem in K. For every fault f' in K', there exists a corresponding fault f in K such that if a sequence I synchronizes K^f to a state q, then I synchronizes $K'^{f'}$ to a state q' equivalent to q.

Next, the result is generalized for any sequence of atomic retiming transformations.

Theorem 3: Let K' be a circuit resulting from a retiming of K, and P be a sequence of arbitrary input vectors of length equal to the maximum number of forward retiming moves across any node in K. For every fault f' in K', there exists a corresponding fault f in K such that if a sequence I synchronizes K^f to a state q, then the sequence $P \cup I$ synchronizes $K'^{f'}$ to a state q' equivalent to q.

Theorem 3 guarantees that by adding a prefix sequence to a synchronizing sequence for a faulty circuit, its corresponding faulty retimed circuit is synchronized to an equivalent state. Thus, after synchronization both faulty circuits have equivalent I/O behavior.

We next show that adding a prefix sequence is necessary for preserving synchronizing sequences on corresponding faulty retimed circuits.

Observation 2: Let K' be a circuit resulting from a retiming of K. Let f in K be the only corresponding fault to f' in K'. Then, a synchronizing sequence that synchronizes the faulty circuit K^f does not necessarily synchronize the faulty circuit $K'^{f'}$.

Example 2: The circuit $N2$ in Fig. 5 is obtained from $N1$ by a single forward retiming move across the single-output AND gate $G1$. The stuck-at-1 fault on line $G1 - Q12$ in $N2$ has one corresponding fault in $N1$ namely, the stuck-at-1 fault on line

$G1-G2$. The sequence $\langle 001, 000\rangle$ synchronizes the circuit $N1$ in the presence of the stuck-at-1 fault on line $G1-G2$ to the state $\{001\}$. However, the same sequence does not synchronize the circuit $N2$ in the presence of the stuck-at-1 fault on line $G1-Q12$. The sequence drives the circuit to the state $\{1x\}$, and hence does not synchronize it. This is due to the fact that the stuck-at-1 fault on line $G1-Q12$ in $N2$ is not space-equivalent to its corresponding single stuck-at fault in the circuit $N1$. However, it is space-equivalent to the multiple stuck-at-1 fault on lines $I1-Q1$ and $I2-Q2$ in $N1$. It is also interesting to observe that the synchronizing sequence $\langle 001, 000\rangle$ is structural-based. Thus, while structural-based synchronizing sequences are preserved under retiming for the corresponding fault-free circuits, they are not preserved for the corresponding faulty circuits. Hence, it is necessary to add the prefix sequence to guarantee synchronization.

C. Preservation of Test Sets

In this subsection, we study the preservation of test sets under retiming. We show that a test set for a retimed circuit can be derived based on a test set for its corresponding original circuit by adding a prefix sequence of a pre-determined number of arbitrary input vectors. The derived test set detects each fault in the retimed circuit corresponding to a fault detectable by the test set in the original circuit.

We have shown, in the previous two subsections, that for every fault f' in a circuit K' resulting from a retiming of K, there exists a corresponding fault f such that if I synchronizes K and K^f to states q and q^f, then the sequence $I' = P \cup I$ synchronizes K' and ${K'}^{f'}$ to states q' and ${q'}^{f'}$ such that q' is equivalent to q and ${q'}^{f'}$ is equivalent to q^f. Here P is a sequence of arbitrary input vectors of length equal to the maximum number of forward retiming moves in K. Thus, this implies that after such synchronization, retiming preserves the I/O behavior of both the fault-free and the faulty circuits. If a test set distinguishes between the states q and q^f and hence detects the fault f in K, then its prefixed version also distinguishes between the states q' and ${q'}^{f'}$ and detects the fault f' in K'. For any given input sequence, such as a set of test vectors, retiming does not alter the sequence of logic values which propagate through each node in the circuit. Retiming can only alter the clock cycle at which the logic values arrive at affected nodes. Because the same logic values are propagated through each node, the same paths are sensitized, and therefore the same (single stuck-at) faults which those vectors detect in the original circuit are detected in the retimed circuit. This concept is formalized in the following theorem.

Theorem 4: Let K' be a circuit resulting from a retiming of K, and P be a sequence of arbitrary vectors of length equal to the maximum number of forward retiming moves across any node in K. For every fault f' in K', there exists a corresponding fault f in K such that if T is a test for f in K, then the sequence $P \cup T$ is a test for f' in K'.

Since we have shown that a test for a fault in a retimed circuit can be derived based on a test for its corresponding fault in the original circuit, this implies that retiming does not introduce any redundant or undetectable faults into the circuit. If a fault in the original circuit is detectable, then all its corresponding faults in the retimed circuit are guaranteed to be detectable. This leads to following important conclusion.

Corollary 1: **Retiming preserves single stuck-at fault testability.**

The significance and the implications of these results on test pattern generation are discussed in Section V.

We next show two situations where without adding a prefix sequence, a test set is not guaranteed to be preserved under retiming.

Observation 3: Let K' be a circuit resulting from a retiming of K. Let f in K be the only corresponding fault to f' in K'. Then, a functional-based test for the fault f in K does not necessarily test the fault f' in K'.

Example 3: Referring to Fig. 3, consider the stuck-at-0 fault on the output of $L2$ and its corresponding stuck-at-0 fault on the output of $L1$. The input vector $\langle 11\rangle$ detects the stuck-at-0 fault on the output of $L1$ since it produces a 1 on the output of the fault-free circuit $L1$ and produces a 0 on the output of the faulty circuit. However, the input vector $\langle 11\rangle$ does not detect the stuck-at-0 fault on the output of $L2$ since it produces an x on the output of the fault-free circuit $L2$ and produces a 0 on the output of the faulty circuit.

Observation 4: Let K' be a circuit resulting from a retiming of K. Let f in K be the only corresponding fault to f' in K'. Then, a structural-based test for the fault f in K does not necessarily test the fault f' in K'.

Example 4: Referring to Fig. 5, the structural-based test sequence $T = \langle 001, 000, 100, 010, 010\rangle$ detects the stuck-at-1 faults on line $I1-Q1$, line $I2-Q2$, line $Q1-G1$, line $Q2-G1$, and line $G1-G2$ in $N1$. However, the test sequence T detects only the stuck-at-1 faults on line $I1-G1$, line $I2-G1$, and line $Q12-G2$ in $N2$. The test sequence T does not detect the stuck-at-1 fault on line $G1-Q12$ in $N2$.

V. Experimental Results

In this section, we demonstrate the effect of retiming on the performance of a structural-based sequential automatic test pattern generator (ATPG). Using fault simulation results, we also demonstrate that the retimed circuits can be tested efficiently and effectively based on the test sets derived for the original circuits.

A. Synthesis of Circuits

Several circuits were synthesized from the MCNC finite-state machine (FSM) benchmarks using the SIS sequential synthesis tool [12]. Table I lists the number of primary inputs, primary outputs and states for the FSM descriptions used to synthesize the circuits. The versions of dk16, pma,

TABLE I
Characteristics of finite-state machines used to synthesize circuits

FSM	PI	PO	States
dk16	3	3	27
pma	9	8	24
s510	20	7	47
s820	18	19	25
s832	18	19	25
scf	27	54	121

TABLE II
Test pattern generation results

	Original				Retimed				
Circuit	#DFF	%FC	%FE	#CPU	#DFF	%FC	%FE	#CPU	CPU Ratio
dk16.ji.sd	5	99.8	100.0	308	19	99.7	100.0	99529	323.1
pma.jo.sd	5	99.4	100.0	791	21	98.8	99.3	183145	231.5
s510.jc.sd	6	98.2	100.0	24507	20	95.3	96.0	405630	16.6
s510.jc.sr	6	94.3	99.3	43060	26	53.9	54.6	415021	9.6
s510.ji.sd	6	99.2	100.0	2918	11	98.8	99.6	165190	56.6
s510.ji.sr	6	98.9	100.0	12460	23	91.4	92.0	343420	27.6
s510.jo.sr	6	96.2	100.0	3822	28	56.5	57.0	1000000	261.6
s820.jc.sd	5	99.4	99.9	1536	14	95.3	96.6	267502	174.2
s820.jc.sr	5	98.7	100.0	1207	9	98.5	99.8	7913	6.6
s820.ji.sr	5	98.2	100.0	8385	8	97.3	100.0	296864	35.4
s820.jo.sd	5	100.0	100.0	1282	22	92.5	93.6	381636	297.7
s820.jo.sr	5	98.6	99.8	1212	13	97.3	98.8	97495	80.4
s832.jc.sr	5	98.4	100.0	1225	27	53.7	56.0	496961	405.7
s832.jo.sr	5	98.1	100.0	1103	15	96.7	99.1	499200	452.6
scf.ji.sd	7	99.6	100.0	17262	20	63.1	63.7	689651	40.0
scf.jo.sd	7	99.6	100.0	16725	23	97.8	97.9	699508	41.8

s510, and scf used employ an explicit reset line.

The SIS command sequence employed followed that suggested in the SIS documentation [13]. For more details on the exact sequence of commands used, the reader is referred to [8]. Since the logic structure of the synthesized circuits depends on the state encoding algorithm and the synthesis script used, multiple (non-retimed) circuits were created from the same FSM descriptions using a variation of the two. Each circuit attained a different area-delay trade-off. The name of each circuit contains multiple fields which reflect the synthesis options employed. The *.j* field denotes the *jedi* state encoding algorithm used: *.jo* represents the output dominant algorithm, *.ji* the input dominant algorithm, and *.jc* a combination of the input and output dominant algorithms. The *.s* field denotes the SIS script used for logic synthesis: *.sd* indicates *script.delay*, and *.sr* indicates *script.rugged*. The presence of a *.re* field indicates that the circuit is a retimed version of the correspondingly named circuit.

B. Test Pattern Generation

The HITEC ATPG [9] was used to measure the increase in the required ATPG CPU time caused by retiming. Table II lists the results of the HITEC ATPG on the synthesized circuits. The columns labeled *#DFF* list the number of DFFs in each circuit, *#CPU* reports the number of DECstation 3100 CPU seconds which HITEC required to attain the levels of fault coverage and fault efficiency reported in the columns labeled *%FC* and *%FE*, respectively, on sequential (non-scan) versions of each circuit. Fault coverage is defined as the percentage of faults which were detected. Fault efficiency is defined as the percentage of faults which were detected or labeled redundant. The rightmost column, titled *CPU Ratio*, lists the ratio of the CPU run times between the retimed and the corresponding original circuits.

The results presented in Table II demonstrate that retiming increases the amount of ATPG time required as well as decreases the achieved levels of fault coverage and fault efficiency. In seven instances, the ratio of CPU time between the retimed and original circuits is more than two orders of magnitude, and in four instances there is a substantial difference between the levels of fault coverage and fault efficiency attained in the original and corresponding retimed circuits. It has been demonstrated in [8] that the increase in complexity by retiming is not specific to the HITEC ATPG, as the effect is shown to exist in two additional state-of-the-art sequential, structural ATPGs.

C. Fault Simulation

Based on the results given in Section IV, a test set for a retimed circuit can be derived based on the test set generated for its corresponding original circuit by adding a prefix sequence of a pre-determined number of arbitrary input vectors. We have found that for the circuits *pma.jo.sd*, *s*510*.jc.sd* and *scf.jo.sd*, there is a maximum of a single forward retiming move across any node when transformed to their retimed versions. Thus, to create test sets for the retimed versions of these circuits, a single arbitrary input vector needs to be prefixed to the test sets generated for the original circuits. For all the remaining circuits, it was found that no additional vectors need to be added as there were no forward retiming moves.

To assess the practical implications of these results, the test sets derived for each of the retimed circuits, based on the test sets generated for the original circuits, were fault simulated on the corresponding retimed circuits using PROOFS [9]. Fault simulation results are shown in Table III. The columns labeled *#Faults* specify the total number of collapsed faults in the original and retimed circuits, respectively. The columns labeled *#UnDet* specify the number of undetected faults resulting from fault simulation of test sets generated for the original circuits on the original circuits, and those derived based on the test sets of the original circuits on the corresponding retimed circuits, respectively.

As can be seen from Table III, the fault simulation results on the original and corresponding retimed circuits are identical in all but eight instances. Any differences between the number of undetected faults in an original and retimed circuit are due to the effect which occurs when a sequential element is added to a line and the fault on that line is mapped to two different faults. For example, referring to Fig. 1(a), retiming the circuit $K2$ to $K1$ causes a fault on $I1$ to be mapped to two separate faults, including a fault on $I1$, and a fault on

TABLE III
FAULT SIMULATION RESULTS

	Original		Retimed	
Circuit	#Faults	#UnDet	#Faults	#UnDet
dk16.ji.sd	587	1	615	1
pma.jo.sd	537	3	587	4
s510.jc.sd	656	12	700	12
s510.jc.sr	628	36	668	36
s510.ji.sd	759	6	769	6
s510.ji.sr	725	8	759	9
s510.jo.sr	612	23	656	25
s820.jc.sd	688	4	706	5
s820.jc.sr	547	7	555	7
s820.ji.sr	547	10	553	10
s820.jo.sd	737	0	777	0
s820.jo.sr	585	8	601	9
s832.jc.sr	552	9	596	11
s832.jo.sr	619	12	639	13
scf.ji.sd	1892	7	1920	9
scf.jo.sd	1955	8	2005	8

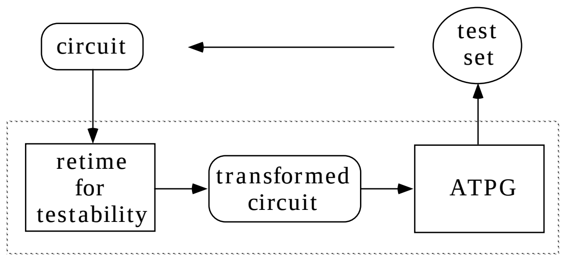


Fig. 6. A new technique for enhancing the performance of ATPG.

the output of $Q0$. The opposite happens when retiming removes a sequential element– two faults are mapped onto a single fault. In such instances, if the test set detects the fault in the original circuit, then it will also detect the two corresponding faults in the retimed circuits. Conversely, if the test set does not detect the fault in the original circuit, then neither of the corresponding faults in the retimed circuit will be detected. This phenomena was found to account for all discrepancies in the number of undetected faults between the original and retimed circuits for the derived test sets. In all other cases, the number of undetected faults for the derived test sets is equal for the original and retimed circuits.

These results demonstrate a powerful technique to offset the dramatic increase in the time required for test generation of retimed circuits. The technique involves targeting for test generation a more easily testable version of the circuit that will be implemented. Fig. 6, originally proposed in [7], summarizes this new technique for enhancing the performance of ATPG. For example, consider the circuit $s510.jo.sr.re$ (given in Table II). The HITEC ATPG is able to achieve only a 56.5% fault coverage in one million CPU seconds on $s510.jo.sr.re$. However, when the circuit $s510.jo.sr.re$ is retimed to the circuit $s510.jo.sr$ by minimizing the number of DFFs, HITEC requires only 3822 seconds to generate a test set which attains a 96.2% fault coverage on $s510.jo.sr$. The test set generated for $s510.jo.sr$ achieves a 96.2% fault coverage when fault simulated on the circuit $s510.jo.sr.re$.

VI. Conclusions

In this paper, we have determined the impact of retiming on the testability of sequential circuits. We have shown that retiming preserves testability with respect to a single stuck-at fault test set by adding a prefix sequence of a pre-determined number of arbitrary input vectors. The significance of this result is illustrated by the fact that performance-driven retiming dramatically increases the run time required by structural sequential ATPGs (an increase of two orders of magnitude in several instances), as well as decreases the levels of fault coverage and fault efficiency attained. We have demonstrated that high fault coverages can be achieved on high performance circuits optimized by retiming in a much less CPU time than if ATPG is attempted directly on those circuits. Testability preservation under retiming thus suggests a new approach for improving the performance of sequential ATPGs and reducing the cost of test pattern generation. Unlike design for testability techniques, this approach does not incur any increase in silicon area or degradation in performance.

Acknowledgment

The authors would like to thank Vigyan Singhal of U.C. Berkeley for his comments on an earlier version of this paper and Nagesh Tamarapalli of McGill University for their helpful discussions.

References

[1] A. El-Maleh, T.E. Marchok, J. Rajski, and W. Maly, "Behavior and testability preservation under the retiming transformation," McGill University Technical Report#94-R3, December, 1994, submitted to *IEEE Trans. CAD.*
[2] A. Ghosh, "Techniques for test generation and verification of VLSI sequential circuits,"Ph.D. Dissertation, U.C. Berkeley Memorandum No. UCB/ERL M91/73, Sept. 1991.
[3] F. Hennie, *Finite state models for logical machines,* New York: John Wiley,1968.
[4] C.E. Leiserson and J.B. Saxe, "Optimizing synchronous systems," *Journal of VLSI and Computer Systems,* vol. 1, pp. 41-67, Spring 1983.
[5] C.E. Leiserson and J.B. Saxe, "Retiming synchronous circuitry," *Algorithmica,* vol. 6, pp. 5-35, 1991.
[6] T.E. Marchok and W. Maly, "Automatic synthesis and the cost of testing," in *Proc. Custom Integrated Circuit Conf.*, pp. 132-135, 1994.
[7] T.E. Marchok, A. El-Maleh, W. Maly and J. Rajski, "Test set preservation under retiming transformation," presented at the *First International Test Synthesis Workshop,* May 18-20, 1994, Santa Barbara, CA.
[8] T.E. Marchok, A. El-Maleh, W. Maly and J. Rajski, "Complexity of sequential ATPG," in *Proc. European Design and Test Conference*, March, 1995.
[9] T.M. Nierman, W.T. Cheng and J.H. Patel, "PROOFS: A fast, memory efficient sequential circuit fault simulator," in *Proc. IEEE Design Automation Conf.*, pp. 535-540, 1990.
[10] T.M. Nierman and J.H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. EDAC*, pp. 214-218, 1991.
[11] C. Pixley and G. Beihl, "Calculating resetability and reset sequences," in *Proc. ICCAD,* pp. 376-379, December 1992.
[12] E. M. Sentovich,K. J. Singh, C. Moon, H. Sajov, R. K. Brayton and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in *Proc. IEEE Int. Conf. Computer Design*, pp. 328-333, 1992.
[13] E. M. Sentovich et al., "SIS: a system for sequential circuit synthesis," U.C. Berkeley Memorandum No. UCB/ERL M92/41, May 1992.