A Fast Sequential Learning Technique for Real Circuits with Application to Enhancing ATPG Performance

Aiman El-Maleh, Mark Kassab, and Janusz Rajski

Mentor Graphics Corporation
8005 S.W. Boeckman Rd.
Wilsonville, OR 97070, USA

1. ABSTRACT

This paper presents an efficient and novel method for sequential learning of implications, invalid states, and tied gates. It can handle real industrial circuits, with multiple clock domains and partial set/reset. The application of this method to improve the efficiency of sequential ATPG is also demonstrated by achieving higher fault coverages and lower test generation times.

2. INTRODUCTION

It is well-known that the performance of a search process can be significantly enhanced by finding necessary assignments and identifying relations between signals. An effective way of identifying necessary assignments and signal relations is by learning. The use of learned information helps the search engine in recognizing value assignment conflicts sooner and pruning the search space, significantly reducing the number of backtracks. Learning relations in a circuit is typically performed by injecting both logic values on a gate and propagating them backward and forward. This can be done either statically in a pre-processing phase [1,2], or dynamically during the search process [3]. Dynamic learning can extract significantly more implications since the learning is done in the presence of assignments that have already been made, which allows the search space to be pruned further. However, this can be very computationally expensive since it is performed multiple times during the search process. Furthermore, the implications learned are only valid as long as backtracking has not occurred beyond the point where learning was performed. In most learning techniques, justification stops when a decision node is reached. Recursive learning [4] attempts to extract more relations by learning implications which would be valid regardless of the assignment made at the decision node. Furthermore, it can be applied statically or dynamically. However, unless the depth of the recursion is restricted, this technique can be even more expensive than the search process. Note that most learning techniques are performed in the combinational logic of the circuit. Only [5] attempts learning of relations by extending the analysis across sequential elements, but the learning is only performed across two time frames.

Learning is used in several areas of computer-aided design, most notably automatic test pattern generation (ATPG) [1-5], redundancy identification [2,6], logic verification [7], and logic optimization [8]. Although this paper focuses on the application of the learning method to ATPG, the proposed method is not restricted to test generation.

Sequential ATPG is a much more complex process than combinational ATPG due to signal dependencies across multiple time frames. Extracting such relations can significantly enhance the performance of sequential ATPG as backtracks across multiple time frames are reduced. Furthermore, it has recently been shown [9] that a key indicator of the complexity of sequential ATPG is the density of encoding of the circuit, which is the ratio of the number of valid states to the total number of states. With circuits that have a low density of encoding, the test generator is more likely to waste a lot of time trying to justify an invalid state. Therefore, extraction of invalid states can also significantly enhance the efficiency of sequential ATPG. Most methods which identify the valid or invalid states in a sequential circuit require the presence of a reset state (e.g., [10]), which is not usually available in real circuits. Some more recent techniques do not require a reset state [11,12]. However, since those methods usually attempt to enumerate all invalid states, they can only be used for small circuits.

In this paper, we present a fast, memory-efficient technique for learning across multiple time frames. We propose the sequential learning mechanism, and its application for enhancing the performance of sequential ATPG by identifying implications, invalid states, and tied values. Its efficiency enables it to be used on very large circuits. Furthermore, it can be used on real circuits including those with multiple clock domains, and partial or full set/reset.

3. SEQUENTIAL LEARNING TECHNIQUE

Sequential learning can identify relations between two nodes at different time frames or in the same time frame. Relations learned between nodes in the same time frame are of more interest since they represent the set of illegal assignments or don’t care set for a circuit, which can be utilized by ATPG, logic optimization or verification. Many more relations can be learned between nodes at different time frames than those between nodes in the same time frame. However, such relations have limited application since they involve the time domain. For an ATPG to take advantage of such relations, it needs to work on a window equivalent to the number of time frames across which the relations hold. The proposed learning technique can learn relations of either type. However, in this paper we present results for relations

learned in the same time frame. Relations are learned between pairs of sequential elements and between gates and sequential elements. Relations between pairs of gates follow directly from relations between gates and sequential elements, and are therefore not extracted.

The sequential learning technique proposed in this paper is mainly based on forward simulation across time frames. This is in contrast to combinational learning techniques, which inject both a 0 and 1 on each gate and perform backward and forward implications across non-decision nodes. Performing the implications both backward and forward can allow learning of relations that may not be learned using only forward or backward implications. This is because a decision node may cease to be a decision node due to other assignments made by implications, allowing further propagation. However, this approach requires the learning mechanism to be performed twice the number of gates in the circuit. Since sequential learning for each gate can traverse the circuit forward and backward across time frames several times, this could make such an approach less attractive for large circuits. Furthermore, the time frames across which learning is to be performed has to be decided and memory has to be allocated a priori.

3.1 Implication Learning

The proposed learning technique is comprised of several steps. First, fanout stems (i.e., nodes that have a fanout greater than one) are identified. Then, for each fanout stem, both a 0 and a 1 are injected and simulated forward across time frames. This step allows the extraction of relations between nodes that are implied by the different values on the same stem based on the contrapositive law. If $A=0 \rightarrow F_1=0$ and $A=1 \rightarrow F_2=0$, then by the contrapositive law, $F_1=1 \rightarrow A=1$ which results in the relation $F_1=1 \rightarrow F_2=0$. This technique is called ***single-node learning***. For example, consider the circuit in Figure 1. This circuit has five fanout stems, namely I_1, I_2, F_1, F_2, and F_3. The forward simulation results for each value on every stem and their corresponding implied gates are shown in Table 1. Simulation is allowed to continue forward for a specified number of time frames. However, the simulation is stopped if the same state is repeated over two consecutive time frames. For example, injecting a 1 on stem F_3 produces the same state after time frame 1 and hence simulation is stopped at time frame 2. Similarly for stem I_2, simulation stops at time frame 4. Relations between nodes at two different time frames can be easily extracted from the simulation results for each stem. For example, the relation $G_1=0$ at $T=i+1 \rightarrow I_2=0$ at $T=i$ is learned based on the stem I_2. Relations between nodes in the same time frame are extracted similarly. For example, the relation $F_6=1 \rightarrow F_4=0$ is extracted based on stem I_2 since $I_2=0 \rightarrow F_6=0$ in time frame 1 and $I_2=1 \rightarrow F_4=0$ in time frame 1 as well. By the contrapositive law, $F_6=1 \rightarrow I_2=1 \rightarrow F_4=0$. Relations extracted between sequential elements based on single-node learning are shown in Table 2. Relations between sequential elements are called *invalid-state relations* since they represent the presence of invalid states. The relation $F_6=1 \rightarrow F_4=0$ represents the set of invalid states $(F_1,F_2,F_3,F_4,F_5,F_6)=(X,X,X,1,X,1)$. Learning of invalid states by implication has the benefit that sets of invalid states are extracted concurrently and their representation is compact. However, invalid states that cannot be represented as implications cannot be learned using this approach.

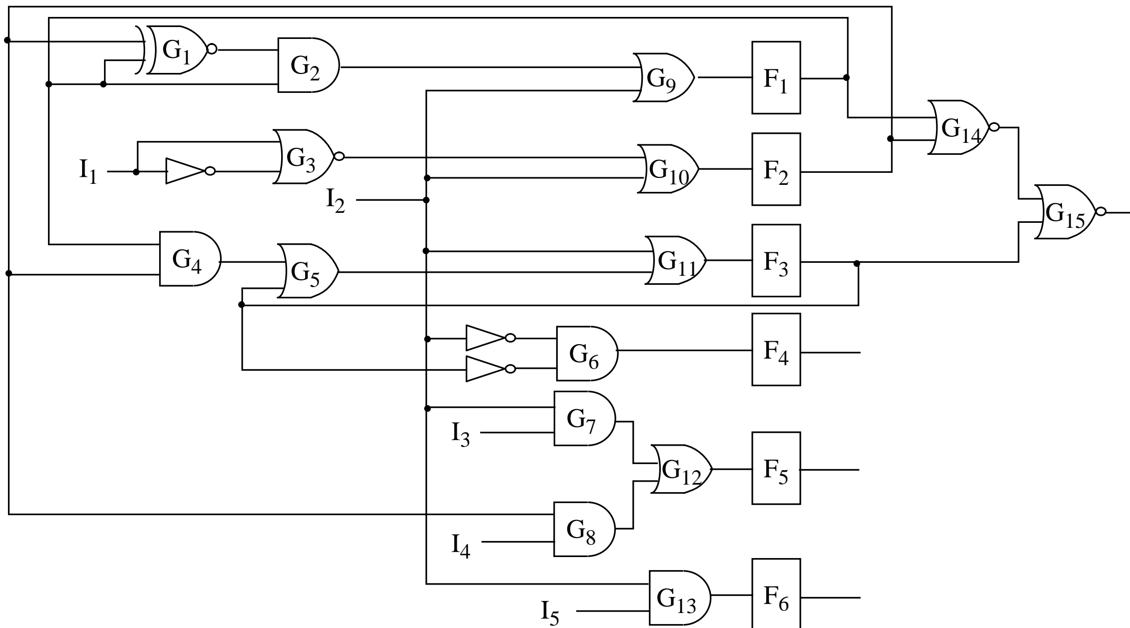

Figure 1: An example illustrating the sequential learning technique

Stem	T=0	T=1	T=2	T=3
$I_1=0$	$G_3=0$	{}	{}	{}
$I_1=1$	$G_3=0$	{}	{}	{}
$I_2=0$	$G_7=0$ $G_{13}=0$	F6=0	{}	{}
$I_2=1$	$G_6=0$, $G_9=1$ $G_{10}=1$ $G_{11}=1$	$F_1=1$, $F_2=1$ $F_3=1$, $F_4=0$ $G_1=1$, $G_2=1$ $G_4=1$, $G_5=1$ $G_6=0$, $G_9=1$ $G_{11}=1$ $G_{14}=0$ $G_{15}=0$	$F_1=1$, $F_3=1$ $F_4=0$, $G_5=1$ $G_6=0$ $G_{11}=1$ $G_{14}=0$ $G_{15}=0$	$F_3=1$, $F_4=0$ $G_5=1$, $G_6=0$ $G_{11}=1$ $G_{15}=0$
$F_1=0$	G2=0, G4=0	{}	{}	{}
$F_1=1$	G14=0	{}	{}	{}
$F_2=0$	G4=0, G8=0	{}	{}	{}
$F_2=1$	G14=0	{}	{}	{}
$F_3=0$	{}	{}	{}	{}
$F_3=1$	$G_5=1$, $G_6=0$ $G_{15}=0$ $G_{11}=1$	$F_3=1$, $F_4=0$, $G_5=1$, $G_6=0$ $G_{11}=1$ $G_{15}=0$	{}	{}

Table 1: Simulation results for stems of the circuit in Figure 1.

Single-Node Relations	Additional Multiple-Node Relations	Additional Gate-Equivalence Relations
$F_6=1 \rightarrow F_4=0$	$F_1=0 \rightarrow F_2=0$	$F_3=0 \rightarrow F_1=0$
$F_6=1 \rightarrow F_3=1$	$F_1=0 \rightarrow F_5=0$	$F_4=1 \rightarrow F_1=0$
$F_6=1 \rightarrow F_2=1$	$F_3=0 \rightarrow F_2=0$	
$F_6=1 \rightarrow F_1=1$	$F_3=0 \rightarrow F_4=1$	
	$F_3=0 \rightarrow F_5=0$	
	$F_4=1 \rightarrow F_2=0$	
	$F_4=1 \rightarrow F_5=0$	
	$F_4=1 \rightarrow F_3=0$	

Table 2: Learned invalid state relations for the circuit in Figure 1.

Single-node learning can only learn relations due to a single stem. However, relations that can be learned due to multiple assignments are missed. This occurs when two or more stems result in a node having the same value. Thus, the contrapositive of the value on the node implies the contrapositive of the values on each of the stems. By injecting the contrapositive values of all the stems and performing forward logic simulation, relations can be identified between the node and other nodes set by the simulation. Injecting the values on multiple stems concurrently usually allows values to propagate further during simulation and hence allows more relations to be extracted. Thus, for each value on a gate, the set of stems that produce this value on the gate are stored. This analysis is called ***multiple-node learning***.

As an example, consider the value 1 on the sequential element F_3 in the circuit shown in Figure 1. This value is set on F_3 at T=1 by each of the stems $I_2=1$ and $F_3=1$, at T=2 by the stem $I_2=1$, and at T=3 by the stem $I_2=1$. Thus, the value $F_3=0$ at T=3 implies that $I_2=0$ at T=3-3=0, $I_2=0$ at T=3-2=1, and $I_2=0$ and $F_3=0$ at T=3-1=2. By injecting those assignments on the stems and performing forward simulation, this results in $F_2=0$ and $F_6=0$ at T=1, $F_2=0$, $F_5=0$, and $F_6=0$ at T=2, and $F_2=0$, $F_4=1$, $F_5=0$, and $F_6=0$ at T=3. Note that the fact that gate G_3 is tied to a 0 is taken advantage of during simulation. Extraction of tied gates will be discussed in the next subsection. This learning leads to the extraction of four invalid state relations, namely $F_3=0 \rightarrow F_2=0$, $F_3=0 \rightarrow F_4=1$, $F_3=0 \rightarrow F_5=0$, and $F_3=0 \rightarrow F_6=0$, the first three of which were not learned by single-node learning. Similarly, multiple-node learning for the other nodes can be performed. Additional invalid state relations learned by multiple-node learning are shown in Table 2.

Taking advantage of equivalent gates in combinational circuits can help values propagate forward during simulation. For example, gates G_2 and G_4 are equivalent combinationally. However, injecting a 0 on F_2 results in $G_4=0$ but does not set $G_2=0$, due to the limitation of three-valued simulation. Knowing that the two gates are equivalent allows the simulator to set G_2 to 0. This can help in identifying extra relations. For example, for the multiple-node learning of $F_3=0$, the value $I_2=0$ at T=0 results in $F_2=0$ and $F_6=0$ at T=1. Then, $F_2=0$ at T=1 results in $G_4=0$ and by equivalence $G_2=0$ at T=1. With $I_2=0$ injected at T=1, this leads to $F_1=0$, $F_2=0$, $F_5=0$, and $F_6=0$ at T=2. Injecting $I_2=0$ and $F_3=0$ at T=2 with the additional assignments on the sequential elements propagating from the previous time frame, this leads to the assignments $F_1=0$, $F_2=0$, $F_4=1$, $F_5=0$, and $F_6=0$ at T=3. Thus, a new relation is extracted between F_3 and F_1 by utilizing gate equivalence in combinational circuits, namely $F_3=0 \rightarrow F_1=0$. Equivalent combinational gates can be efficiently identified based on parallel pattern simulation techniques. Multiple-node learning for the other nodes can be performed similarly. Additional invalid state relations learned by multiple-node learning taking advantage of gate equivalence are shown in Table 2.

It is worth mentioning that the multiple-node learning technique can identify relations that are not learned by techniques based on backward/forward techniques. This is illustrated by the example shown in Figure 2. From the figure, it can be seen that each of the stems $I_2=0$ and $I_3=0$ at T=0 imply $G_9=1$ at T=1. Thus, $G_9=0$ at T=1 implies that $I_2=1$ and $I_3=1$ at T=0, which implies that $F_2=0$ at T=1. Thus, the relation $G_9=0 \rightarrow F_2=0$ is learned. Such a relation cannot be extracted by injecting a 0 or 1 on G_9 and performing backward/forward implications.

3.2 Tie Gate Learning

In addition to learning implications, this learning technique can effectively identify tie gates. A tie gate is one that can only assume a single known value. Tied gates can be tied combinationally or sequentially. A combinational gate tied to the value $v \in \{0, 1\}$ will always be set to the value v and cannot be set to the value $\bar{v}$. A sequential gate tied to the

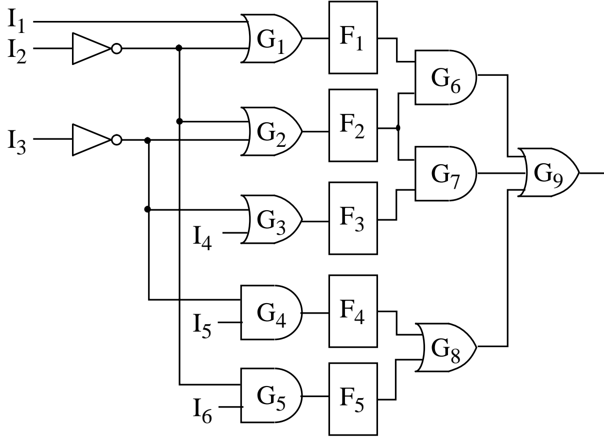

Figure 2: An example illustrating extraction of relations by multiple-node learning that are not extracted by backward/ forward approaches.

value $v \in \{0,1\}$ will always be set to the value v for each of the reachable states. If the circuit is powered up in one of the unreachable states, then the sequentially tied gate can initially be set to $\bar{v}$ and after some number of cycles will be set to the value v. Under three-valued simulation, a sequentially tied gate will only get the value v once it is set to a known value. It is worth noting that sequentially tied gates are by the definition in [13] c-cycle redundant for some cycle c. Thus, identification of tied gates can help in identifying untestable faults in the circuit and also in optimizing the design by removing the redundant gates.

Our technique identifies tied gates based on both single- and multiple-node learning techniques. If both a value of 0 and a value of 1 on a stem produce the same value v on a node G at the same time frame, then this implies that node G is tied to the value v and hence the fault stuck-at-v is untestable. For example, consider the circuit in Figure 1. Both I_1=0 and I_1=1 imply that G_3=0. Thus, gate G_3 is tied to 0. This is considered a combinational tie since it is learned at T=0. Extraction of tied gates based on this criterion is performed after the single-node learning phase, so that the multiple-node learning phase can take advantage of learned tied gates.

Tied gates can also be learned during the multiple-node learning phase. This is illustrated by the example given in Figure 1. From Table 1, it can be seen that by the contrapositive law, G_{15}=1 implies I_2=0 at T=0, I_2=0 at T=1, I_2=0 and F_3=0 at T=2, and F_3=0 at T=3. Injecting those assignments in their respective time frames and simulating forward, this implies that F_2=0 and F_6=0 at T=1, F_1=0, F_2=0, F_5=0, and F_6=0 at T=2, and F_1=0, F_2=0, F_4=1, F_5=0, and F_6=0 at T=3. The assignments F_1=0 and F_2=0 at T=3 imply that G_{14}=1 which implies that G_{15}=0. Thus, we get a conflict which indicates that G_{15} cannot be assigned the value 1, and hence is learned to be sequentially tied to 0. It is also possible to learn that gate G_{15} is a tie gate in a different way during logic simulation. For example, if G_{14} is a stem, then we will have the implication $G_{15}=1 \rightarrow G_{14}=0$ at T=3. However, from the multiple-node learning we described above, simulation of the injected values results in G_{14}=1 at T=3, which is again a conflict indicating that G_{15} is a tie gate. Thus, conflicts occurring during logic simulation indicate that the target gate for which learning is performed is a tie gate. It should be observed that this gate would not have been learned to be a tie without taking advantage of the previously learned tie gate G_3 and the equivalence relation between gates G_2 and G_4.

3.3 Practical Issues

Industrial circuits are often featured with multiple clock domains, multiple-phase clocks, multiple-port latches, and partial set/reset. Such practical issues require special consideration during learning.

3.3.1 Multiple-port Latches

Multiple-port latches have several ports through which their values can be assigned. To guarantee that the extracted relations during learning are valid, values are not allowed to propagate across such latches. This is because the extracted relations would otherwise only be valid with respect to a certain port.

3.3.2 Multiple clock domains

To extract learned relations that are valid regardless of temporal information and the clock domain used, sequential elements are classified into classes where each class consists of those that are driven by the same or equivalent clock and at the same phase. A clock and a gated version of that clock are considered as separate clocks. Then, learning is performed for each class separately. Latches and flip-flops are also classified into different classes even if they are driven by the same clock and at the same phase, since their different capture times may invalidate the learned relations.

3.3.3 Set/Reset handling

Values are not allowed to propagate across sequential elements that have both unconstrained set and reset lines. This is because the learned information will be invalidated by the set/reset behavior since both a 0 and 1 can be obtained on the sequential element regardless of the value being propagated. However, if a sequential element contains either an unconstrained set or reset line, then a value is allowed to propagate across the sequential element if the value is the same as what could be produced by the set or reset line. This guarantees that the extracted relations cannot be invalidated by the set/reset behavior and that all the extracted relations are valid.

4. ENHANCING ATPG PERFORMANCE WITH LEARNED DATA

Implication learning can enhance ATPG performance by identifying conflicts early in the search process and by eliminating decision nodes whose values are implied. Implication relations can be used as known-value implications or forbidden-value implications. Forbidden-value implications are useful in identifying illegal assignments. However, they do not eliminate implied

decision nodes as known-value implications do. We illustrate this by the example shown in Figure 2. To detect the stuck-at-1 fault on G_9, it is necessary to set $G_9=0$ to excite the fault. An implication relation exists between G_9 and F_2 namely, $G_9=0 \rightarrow F_2=0$. Without using the implication, G_6 and G_7 become decision nodes with two solutions each, $F_1=0$ or $F_2=0$ for justifying a 0 on G_6, and $F_2=0$ or $F_3=0$ for justifying a 0 on G_7. If a solution other than $F_2=0$ is selected, this results in additional unnecessary requirements to detect the fault. By taking advantage of the known-value implication, $G_6=0$ and $G_7=0$ are longer decision nodes and hence the search space is pruned. It should be observed that if the implication is used as forbidden-value implication i.e., $G_9=0 \rightarrow F_2=\bar{1}$, then no benefit is gained as $G_6=0$ and $G_7=0$ remain decision nodes.

While known-value implications are more useful than forbidden-value implications, they could result in unnecessary requirements. To illustrate this, consider the example in Figure 1. Suppose that we need to justify $F_6=1$. By known-value implication, this would require the ATPG to justify $F_4=0$, $F_3=1$, $F_2=1$, and $F_1=1$. If any of those requirements is targeted by the ATPG before the requirement $F_6=1$, this can complicate the ATPG further and result in unnecessary requirements. In addition to causing unnecessary requirements, known-value implications can cause the ATPG to declare testable faults as untestable, as demonstrated in [15].

To eliminate the unnecessary requirements caused by learned implications, we propose the following solution. Learned implications are used as forbidden-value implications. However, forbidden values are implied forward and backward dynamically during the ATPG process similar to known-value implications. Forbidden 0 ($\bar{0}$) is implied as 1, and forbidden 1 ($\bar{1}$) is implied as 0. Then, when a value on a decision node needs to be justified, the inputs of the node are examined and the input with the forbidden non-controlling value is selected. This provides the same benefit that known-value implications provide without justifying unnecessary values. However, while known-value implications can eliminate decisions forward, forbidden-value implications do not. To illustrate this, consider the example in Figure 2 again. Using the implication as forbidden-value implication, we have $G_9=0 \rightarrow F_2=\bar{1}$. To justify a 0 on G_6, the input with a $\bar{1}$ assignment is selected, which is F_2. Then, a 0 on F_2 is justified which is what a known-value implication would have required. However, if justifying a 0 on F_2 fails, then in the case of forbidden-value implications backtracking will occur and the other choices for G_6 and G_7 will be tried and will fail too. In this case, known-value implications will be more effective. Considering the example in Figure 1, justifying $F_6=1$ would not require justifying any other value since no assignments are required on F_4, F_3, F_2, and F_1.

Since the use of each of the known- and forbidden-value implications has relative advantages and disadvantages, neither method is expected to consistently perform better. However, it is expected that the use of forbidden-value implications can be enhanced to capture the full benefit of known-value implications without its associated problems.

5. Experimental Results

Experiments using the proposed learning method were conducted on several sequential circuits, including ISCAS 89 and 93 benchmark circuits, 4 retimed circuits, and 3 industrial circuits. The retimed circuits were chosen as it was shown that ATPG can be particularly inefficient on this class of circuits due to the large number of invalid states introduced by retiming [9]. The 4 retimed circuits chosen were those with the lowest density of encoding, and which consequently had the highest APTG complexity when run on HITEC [14]. The industrial circuits were used to demonstrate the method on real circuits with set/reset and multiple clock domains, and to show the high efficiency of the method when handling very large circuits. All experiments were run on a 167 MHz Sun Ultra 1.

5.1 Sequential Learning Results

Sequential learning is performed with simulation allowed to propagate a maximum number of 50 time frames. In the results shown in Table 3, the relations reported are those learned sequentially; the relations which can be learned in the combinational logic are excluded to isolate what can only be extracted by sequential learning. The learned relations between pairs of FFs and between gates and FFs are reported. The efficiency of the sequential learning technique is clearly demonstrated as for the largest circuit, which has over 680,000 gates, it requires less than 7 minutes.

To illustrate the effectiveness of our approach at identifying tie gates, we compare the number of untestable faults identified due to tie gates with those identified by FIRES [13]. FIRES uses sequential analysis to identify faults that cannot be activated or cannot be propagated due to conflicts on stems. So, it targets a more general class of untestable faults than those which cannot be activated or propagated due to tie gates. However, our method identifies untestable faults as a by-product of learning tie gates during the sequential learning process, and is not targeted for this purpose. Table 4 shows the number of untestable faults identified due to tie gates and those identified by FIRES. In three circuits, namely s5378, s3330, and s38417, more untestable faults are identified with tie gate information than FIRES was able to identify.

5.2 ATPG Results

Sequential ATPG experiments were used to demonstrate the benefit of the learned information in enhancing the efficiency of test generation. The experiments were run with and without the use of sequential learning. With sequential learning enabled, two scenarios were used: using the relations between signals as forbidden-value implications as proposed, or as known-value implications. Note that all the ATPG experiments performed make use of combinational learning. Therefore, the difference in the results reported is only due to the use of sequential learning.

For the smaller benchmark circuits in Table 3 (s382 through

Circuit	FFs	Gates	Relations		CPU (s)
			FF-FF	Gate-FF	
s382	21	158	9	37	0.06
s386	6	159	8	135	0.04
s400	21	164	12	47	0.07
s444	21	181	11	69	0.08
s641	19	377	36	197	0.04
s713	19	393	36	216	0.06
s953	29	424	145	1870	0.78
s967	29	395	126	1437	0.43
s1196	18	529	8	44	0.07
s1238	18	508	9	48	0.07
s1269	37	569	30	232	0.06
s1423	74	657	4	251	0.16
s3330	132	1789	367	1764	1.30
s3384	183	1685	31	48	0.19
s4863	104	2342	256	17398	4.15
s5378	179	2779	250	2233	6.42
s6669	239	3080	24	1603	0.39
s9234	228	5597	416	7321	4.38
s13207	638	7951	1566	35093	23.08
s15850	597	9772	1516	29378	42.04
s38417	1636	22179	1554	46981	30.24
s38584	1452	19253	2320	32372	41.93
s510jcsrre	26	243	127	891	0.10
s510josrre	28	243	50	484	0.07
s832jcsrre	27	195	125	743	0.11
scfjisdre	20	764	22	1980	0.56
indust1	460	8693	118	6774	2.74
indust2	7068	63156	2069	36397	24.31
indust3	15689	681595	8016	186930	403.30

Table 3: Sequential learning experiments

Circuit	Untestable faults	
	Tie Gates	FIRES
s5378	441	367
s3330	232	161
s9234	61	284
s13207	182	893
s15850	69	332
s38417	192	147
s38584	538	1437

Table 4: Comparison of untestable faults identified due to tie gates with those identified by FIRES[13].

s1269), no results were reported since the ATPG was originally efficient and there was no benefit from using the sequential learned data. Table 5 reports the test generation results on the larger benchmark circuits which the ATPG tool has difficulty with. The experiments were run in two stages, with different backtrack limits. This allows a better understanding of the trade-offs between ATPG results and CPU time. From the results summarized in the table, it can be seen that in most cases, sequential learning can increase the number of detected faults and those identified as untestable, as well as decrease the ATPG time. For example, based on forbidden-value implications, s5378 with a backtrack limit of 1000 achieves over 7% higher test coverage (fault coverage excluding untestable faults) with 42% less CPU time. Even using the 1000 abort limit and spending over 15 times more time, the ATPG without sequential learning fails to achieve the test coverage that ATPG with learning obtains using a 30 abort limit. Another example is s4863, in which the use of known-value implications gives approximately 4% higher test coverage in less than one-seventh the ATPG time.

For the retimed circuits, higher fault coverage and lower ATPG times were obtained using sequential learning. For example, based on forbidden-value implications, s832jcsrer with a backtrack limit of 1000 achieves over 31% higher test coverage in less than half the CPU time, compared to no learning. Note that for retimed circuits, a solution was proposed in [16] for enhancing ATPG performance in which the retimed circuit is transformed prior to ATPG to increase the density of encoding. Although that solution is very effective at enhancing ATPG results, the sequential learning solution has the advantage that it is not restricted to this class of circuits.

As predicted, experimental results indicate that neither known-value implications nor the current use of forbidden-value implications consistently performs better.

In a few cases, the use of sequential learning results in lower fault coverage. For example, for the circuit s510josrre, using forbidden-value implications resulted in fewer detected faults than without learning. This was analyzed to be caused by random effects. In ATPG, after a test sequence is generated for a fault, the sequence is fault-simulated and all detected faults are dropped. Therefore, it is possible to detect a fault which ATPG would fail to find a test for. For the cases where there were fewer faults detected with learning, the additional faults which the original ATPG detected were targeted individually. In all cases, the ATPG failed to find a test for them, proving they were detected by fault simulation of tests generated for other faults.

6. Conclusions

We have presented a novel and very efficient method for sequential learning, which can be used to identify implications, invalid states, and tied gates. While most learning techniques published in the literature perform learning in the combinational logic, the method presented also learns information which can only be extracted by performing the analysis across memory elements. The method can be used on large industrial circuits as it is extremely fast, and capable of handling real circuit issues such as multiple clock domains, and partial or full set/reset.

The learned information can be used in a variety of design automation tools, including test pattern generation, redundancy identification, logic verification, and logic optimization. The experiments performed focus on demonstrating the use of sequential learning to enhance sequential ATPG efficiency. Although the technique can also perform combinational learning, the focus of the experiments was to isolate the effect of sequential relations and demonstrate the benefit of doing the learning sequentially. It can be observed for most circuits that when ATPG uses the learned information, the number of detected faults and those identified as untestable is higher, and the

Circuit	Total faults	Backtrack limit	No learning			Forbidden values			Implications		
			Det	Untest	CPU(s)	Det	Untest	CPU(s)	Det	Untest	CPU(s)
s1423	1515	30	1329	19	685	1375	19	676	1385	19	716
		1000	1343	19	4029	1381	19	3711	1395	19	3674
s3330	2870	30	2123	464	368	2124	542	276	2124	359	701
		1000	2124	484	6801	2124	562	4848	2124	359	10572
s3384	3380	30	3141	1	500	3155	1	420	3155	1	414
		1000	3211	1	7235	3183	1	5807	3183	1	5784
s4863	4764	30	4495	0	250	4600	126	100	4615	126	103
		1000	4566	0	2069	4612	126	350	4623	126	271
s5378	4603	30	3411	289	1761	3540	630	1370	3519	642	1440
		1000	3454	432	21012	3564	646	12095	3561	646	15995
s6669	6684	30	6574	0	189	6624	0	156	6632	0	197
		1000	6596	0	2025	6654	0	969	6658	0	1168
s13207	9815	30	892	8887	14341	892	8896	16154	895	8903	15562
		1000	892	8908	15362	892	8916	16915	895	8913	16298
s510jcsrre	668	30	309	9	134	309	9	148	309	9	144
		1000	477	11	1775	563	11	1065	556	11	1134
s510josrre	656	30	179	4	142	179	4	150	179	4	157
		1000	556	4	1126	487	4	1745	565	4	1077
s832jcsrre	596	30	343	0	65	331	0	74	266	0	100
		1000	369	0	1496	556	0	659	525	0	850
scfjisdre	1920	30	507	6	680	503	6	811	503	6	764
		1000	1700	6	7281	1714	6	7929	1746	6	6645

Table 5: ATPG experiments with and without sequential learning

ATPG time is significantly reduced. The benefit of sequential learning can be particularly observed on retimed circuits, which have a large number of invalid state relations that can significantly increase ATPG complexity.

7. ACKNOWLEDGEMENTS

The authors would like to thank Wu-Tung Cheng, Rob Thompson, and Srinivas Patil of Mentor Graphics for their useful discussions.

8. REFERENCES

[1] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Transactions on Computer-Aided Design*, pp. 126-136, January 1988.

[2] J.-K. Zhao, E. M. Rudnick, and J. H. Patel, "Static Logic Implication with Application to Redundancy Identification," in *Proc. VLSI Test Symposium*, pp. 288-293, April 1997.

[3] W. Kunz and D. K. Pradhan, "Accelerated Dynamic Learning for Test Pattern Generation in Combinational Circuits," *IEEE Transactions on Computer-Aided Design*, pp. 684-694, May 1993.

[4] W. Kunz and D. K. Pradhan, "Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Circuits," in *Proc. International Test Conference*, pp. 816-825, September 1992.

[5] E. Auth and M. H. Schulz, "A Test-Pattern-Generation Algorithm for Sequential Circuits," in *IEEE Design and Test of Computers*, pp. 72-85, June 1991.

[6] M. A. Iyer and M. Abramovici, "FIRE: A Fault-Independent Combinational Redundancy Identification Algorithm," *IEEE Transactions on VLSI Systems*, pp. 295-301, June 1996.

[7] W. Kunz, "HANNIBAL: An Efficient Tool for Logic Verification Based on Recursive Learning," in *Proc. International Conference on Computer-Aided Design*, pp. 538-543, November 1993.

[8] W. Kunz and P. R. Menon, "Multi-Level Logic Optimization by Implication Analysis," in *Proc. International Conference on Computer-Aided Design*, pp. 6-13, 1994.

[9] Thomas Marchok, Aiman El-Maleh, Wojciech Maly, and Janusz Rajski, "A Complexity Analysis of Sequential ATPG," *IEEE Transactions on Computer-Aided Design*, Vol. 15, pp. 1409-1423, Nov. 1996.

[10] H. Touati, H. Savoj, B. Lin, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Implicit State Enumeration of Finite State Machines using BDD's," in *Proc. International Conference on Computer-Aided Design*, pp. 130-133, November 1990.

[11] D. E. Long, M. A. Iyer and M. Abramovici, "Identifying Sequentially Untestable Faults Using Illegal States," in *Proc. VLSI Test Symposium*, pp. 4-11, 1995.

[12] H.-C. Liang, C. L. Lee, and J. E. Chen, "Invalid State Identification for Sequential Circuit test Generation," in *Proc. Asian Test Symposium*, pp. 10-15, 1996.

[13] M. A. Iyer, D. E. Long, and M. Abramovici, "Identifying Sequential Redundancies Without Search," in *Proc. VLSI Test Symposium*, pp. 4-11, 1995.

[14] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," in *Proc. European Design Automation Conference*, pp. 214-218, 1991.

[15] Wu-Tung Cheng, Aiman El-Maleh, Rob Thompson, Don Ross, and Janusz Rajski, "The Pitfalls of Necessary Assignments", *Fourth International Test Synthesis Workshop*, May 1997, Santa Barbara, CA.

[16] Aiman El-Maleh, Thomas Marchok, Janusz Rajski, and Wojciech Maly, "Behavior and Testability Preservation Under the Retiming Transformation," *IEEE Transactions on Computer-Aided Design*, Vol. 16, pp. 528-543, May 1997.