Identifying Untestable Faults in Sequential Circuits

HSING-CHUNG LIANG
CHUNG LEN LEE
National Chiao Tung University
JWU E. CHEN
Chung-Hua Polytechnic Institute

This article proposes an efficient method to identify untestable faults in sequential circuits. It uses a controllability calculation and symbolic simulation procedure that propagates the characteristics of unknown initial flip-flop states throughout the circuit. Identifying flip-flops that cannot be initialized and circuit lines that cannot be justified to definite values, this process classifies and identifies four types of untestable faults. Experimental results show that it improves the efficiency of a test generation system.

UNTESTABLE FAULTS in a circuit are faults that no input patterns can detect. They cause difficulty in test generation, especially in generating tests for sequential circuits. To improve test generation performance, we want to identify these untestable faults before rather than during the test generation process.

We classify untestable faults into two types: combinationally redundant and sequentially untestable. Redundant circuit lines cause combinationally redundant faults, and we usually identify such faults within one time frame using a test generation process.

Sequentially untestable faults, however, can be further classified into sequentially redundant and irredundant-but-untestable faults. These sequentially redundant faults do not affect the original circuit's function. That is, when we apply an input sequence, the fault-free and faulty circuits have equivalent output responses. Irredundant-but-untestable faults are usually caused by the inability to initialize flip-flops in fault-free or faulty circuits, which hinders the test generator's search for a test sequence. During test generation, identifying sequentially untestable faults usually needs many time frames, and sometimes may not succeed within the limited time available. The search for such faults consumes time and reduces the efficiency of the test generation process. Several researchers have developed techniques for identifying these faults; see the Identifying faults box for more information.

We propose a method that quickly identifies untestable faults for the single observation time strategy. Without extracting a combinational model from the sequential circuit or using a test generator, we identify untestable faults by calculating the controllability and symbolically simulating a circuit to obtain the defined characteristics. According to the controllability of each circuit line and the characteristics of the simulated fault-free and faulty circuits, we classify circuit lines into two types of uncontrollable lines. Then we identify invalid states and four types of defined, untestable faults. We incorporated this untestable-fault identification method into a se-

0740-7475/95/$04.00 © 1995 IEEE

Identifying faults

Researchers have proposed several methods to identify combinationally redundant faults,[1-5] some based on analyzing the structure[1] or the circuit's controllability and observability.[2] The method of Abramovici, Miller, and Roy[3] relied on test-covering relations among faults. Other methods used the processes of test generation and fault simulation repeatedly[4] or in one pass[5] to find redundant faults. A sequential-circuit test generator can use these methods to identify combinationally redundant faults in sequential circuits.

For sequentially untestable faults, Cheng[6] adopted an approach defining a feedback-free circuit model in which he cut several feedback lines from the original circuit to obtain a combinational model. He found so-called feedback-free sequential redundancies from the combinational model using a test generator. Results depended on the selected feedback lines.

Moondanos and Abraham[7] used formal verification techniques to identify redundant faults, taking advantage of the fact that faulty circuits with redundant faults and the fault-free circuit have equivalent state tables. This method is complicated, especially for large circuits, since it requires building state transition tables for the good circuit, and (for each fault modeled) the faulty circuits. Agrawal and Chakradhar[8] used a combinational test generator to target certain faults in an iterative array model of finite length derived from a sequential circuit.

Recently, Iyer and Abramovici[9] proposed a method based on the simple concept that a fault requiring an illegal combination of values as a necessary condition for its detection is untestable. They also expanded sequential circuits into a finite number of time frames and, from this combinational iterative-array model, used implications to find faults whose detection requires conflicts on certain lines in the circuit. Pomeranz and Reddy[10] distinguished the testing strategies with no reset states to single and multiple observation time strategies. Single observation time strategy finds an input sequence to synchronize both the fault-free and faulty circuits to have different output responses at a specific time. On the other hand, multiple observation time strategy[10] also finds an input sequence, uses multiple observations of the circuit response, and detects a fault if the fault-free and faulty responses differ at some time unit. So, multiple observation time strategy may detect some faults identified untestable by single observation time strategy, especially when the fault-free or faulty circuits have no synchronizing input sequences. The method[10] still required a test generator and showed results only for small benchmark circuits and for faults that a single observation time test generator failed to find.

quential-circuit test generator based on the backward-justification[11] algorithm. Experimental results show that our method quickly identifies more untestable faults than other methods[6,8,9,11-13] and, as a result, greatly enhances the test generator's efficiency.

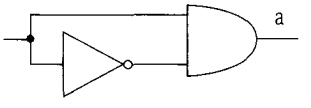


Figure 1. *Circuit where line a cannot have definite value 1.*

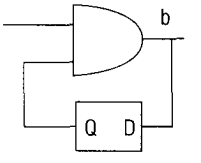


Figure 2. *Circuit where line b cannot have definite value 1; Q and D are the output and input signals for a D flip-flop.*

Uncontrollable lines

For a sequential circuit with unknown initial states, under single observation time strategy, there may exist some circuit lines that cannot have definite values 0 or 1 no matter what input sequences we apply to the primary inputs. These are uncontrollable lines, for which there are two cases:

- Case 1—The circuit line is combinationally redundant. For example, in Figure 1, line a cannot have the definite value 1 since it is combinationally redundant at 0.
- Case 2—The unknown initial flip-flop states always determine the value of the circuit line. Figure 2 shows an example where line b cannot have the definite value 1 if the flip-flop's initial state is unknown.

We can identify case-1 lines in one time frame because they are combinationally redundant. But for case-2 lines, we may need more than one time frame. During test generation, especially during justification of a circuit line to some definite value, the test generator usually wastes time justifying the uncontrollable values on case-2 lines. Therefore, it is desirable to obtain information on these uncontrollable lines before test generation. In addition, as we will show later, these uncontrollable lines help identify invalid states and untestable faults.

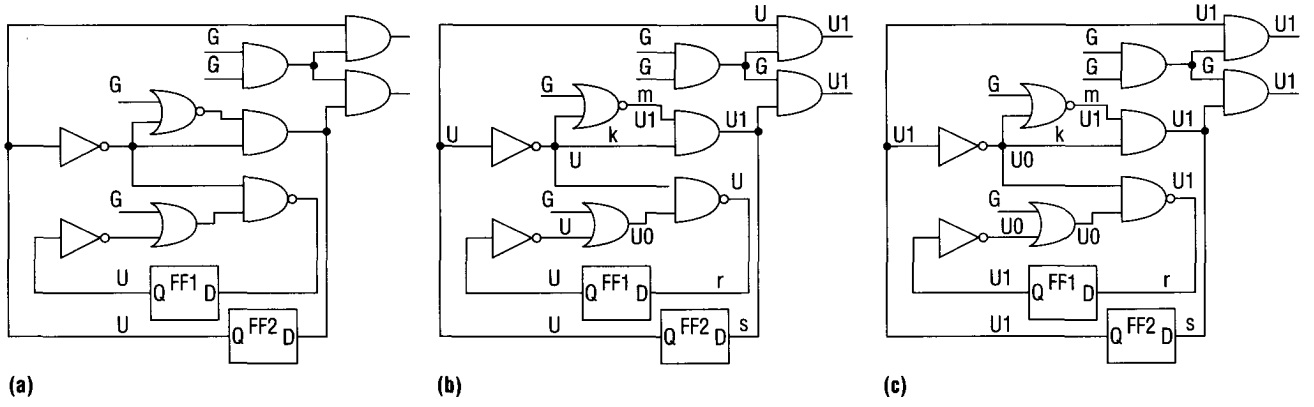


Figure 3. *Finding circuit line characteristics: initial assignment* ***(a)****, propagating characteristics in the first time frame* ***(b)****, and after all circuit lines reach their final, stable characteristics* ***(c)****.*

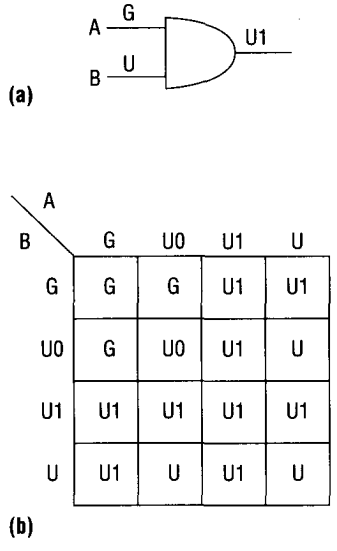


Figure 4. *A two-input AND gate* ***(a)*** *and its characteristic propagation table* ***(b)****.*

Controllabilities and characteristics. Controllabilities $CY1(i)$ and $CY0(i)$ of circuit line i are measures of this line that we can control to be 1 or 0 from inputs. We calculate controllabilities for a sequential circuit using the following procedure.

First, assign 0.5 to the $CY1$ and $CY0$ of all the primary inputs and all the output lines of flip-flops. Second, calculate the controllability of each gate from a lower to higher level using

$$CY1(i) = \prod_{k \in I_i} CY1(k)$$

$$CY0(i) = 1.0 - CY1(i)$$

where I_i is the set of its input lines, and if the gate is an AND gate. For an OR gate, use

$$CY0(i) = \prod_{k \in I_i} CY0(k)$$

$$CY1(i) = 1.0 - CY0(i)$$

Third, given that i_m and o_m are the input and output lines of flip-flop m, if both

$$\sum_{m \in FFs} \left| CY1(i_m) - CY1(o_m) \right| < \varepsilon$$

$$\sum \left| CY0(i_m) - CY0(o_m) \right| < \varepsilon$$

then stop calculating. (ε is a preset small value, say 0.01). If not, propagate the controllability of the input line for each flip-flop to its output line and calculate the controllability of the next frame.

Four symbols represent the circuit line states and identify the uncontrollable lines of a circuit. Under the single observation time strategy, for a sequential circuit with unknown initial states, characteristic $CH(i)$ of a circuit line i describes our ability to justify the line to a definite value during test generation. $CH(i)$ can be one of the following four values:

- U if we cannot justify either values 1 and 0 on line i; that is, line i is always in an unknown state
- U1 if we cannot justify the value 1 on line i
- U0 if we cannot justify the value 0 on line i
- G if we can justify both values 1 and 0 on line i

We use Figure 3's example circuit to demonstrate the process of finding circuit line characteristics. In Figure 3a, we initially assign all the primary inputs G since they can have the definite values 1 and 0. At the same time, we assign all the flip-flop outputs U to reflect their initially unknown states. To find the characteristic of each circuit line, we propagate (or simulate) these initial characteristics through the circuit one time frame after another until the circuit reaches a stable condition, as explained later.

To propagate characteristics, we define characteristic propagation tables for each type of gate. As an example, Figure 4 shows the table for a two-input AND gate. Figure 3b shows the result of propagating characteristics through the circuit for the first time frame. In the fig-

ure, input characteristic U1 of flip-flop FF2 is different from its output characteristic U, which means that the propagation is not yet stable. The U1 of FF2 needs to be further propagated in the next time frame until all the circuit lines reach their final stable values. The example circuit of Figure 3 needed three time frames of propagation to achieve a stable condition. Figure 3c shows the final results

Uncontrollable lines of fault-free and faulty circuits. From the definition of characteristic *CH* and the obtained controllability and characteristics, we identify uncontrollable circuit lines. According to controllability, lines with $CY1 = 0$ cannot have definite value 1, and lines with $CY0 = 0$ cannot have definite value 0 because these values cannot be controlled from primary inputs. They are equivalent to lines with characteristics U1 and U0.

That is, lines for which $CH(i)$ equals U cannot have the definite values 1 or 0 no matter what input sequence we apply to the circuit. Flip-flop values, rather than primary input, control these lines. Since the associated flip-flops are in unknown states, these lines cannot have definite values, either. Thus, we define two types of uncontrollable lines

- 1-uncontrollable lines have $CY1$ equal 0 or $CH(i)$ equal U1 or U
- 0-uncontrollable lines have $CY0$ equal 0 or $CH(i)$ equal U0 or U

For example, in Figure 3c, line m is a 1-uncontrollable line, and line k is a 0-uncontrollable line.

The characteristics and uncontrollable lines just described are for fault-free circuits. We can apply a similar process to identify the characteristics and uncontrollable lines in a faulty circuit with injection of a target fault. For example, considering the stuck-at-1 fault on line m in Figure 3, we can apply a similar process to derive the final characteristics of each line in the faulty circuit, as shown in Figure 5.

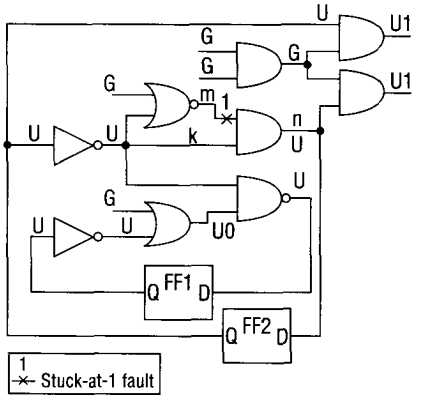


***Figure 5.** Final characteristics of Figure 3's faulty circuit with the stuck-at-1 fault on line m.*

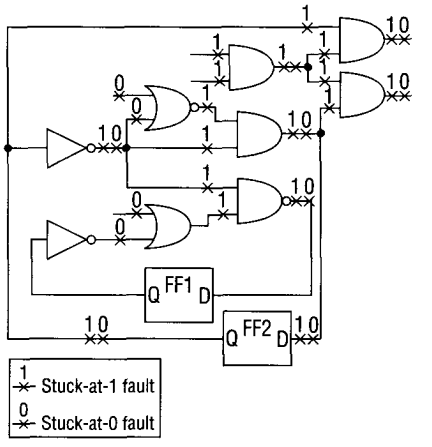


***Figure 6.** Total faults of the example circuit after collapsing the equivalent faults.*

Identifying invalid states and untestable faults

We use the characteristics and uncontrollable lines in fault-free and faulty circuits to identify invalid states and untestable faults. Invalid states are states that cannot be reached no matter what input sequences we apply. We classify untestable faults into four types: unexcitable, unpropagatable, undrivable, and unsensitizable.

Initialization and invalid states. We cannot initialize a sequential circuit if all the flip-flops cannot reach definite values for any applied input sequence. We derive the following rules to identify whether or not we can initialize a circuit and also to identify the invalid states in it:

- Rule 1—We cannot initialize a sequential circuit if all the flip-flop input lines are both 1-uncontrollable and 0-uncontrollable lines.
- Rule 2—For a sequential circuit, if the input line of one flip-flop is 1-uncontrollable (or 0-uncontrollable), the states of this flip-flop that are composed of 1 (or 0) are invalid states.

In Figure 3c, we know from rule 2 that states 10 and 11 are invalid states because line r is 1-uncontrollable. States 01 and 11 are invalid states because line s is 1-uncontrollable.

We will use the circuit of Figure 3 as an example to describe how to use information on uncontrollable lines and characteristics to identify the four types of untestable faults. We collapse the equivalent faults[14] to obtain 30 faults for this circuit (Figure 6).

Unexcitable faults. A fault is unexcitable if the associated circuit line cannot be set to the fault-free value opposite the fault. From the definitions for 1- and 0-uncontrollable lines, we have the following rule:

- Rule 3—Stuck-at-1 faults on 0-uncontrollable lines and stuck-at-0 faults on 1-uncontrollable lines are unexcitable faults (UEFs).

Of the 30 faults in Figure 6, we identified ten UEFs using rule 3 and show them in Figure 7 on the next page.

Unpropagatable faults. A fault is unpropagatable if there exist no input sequences to propagate the fault. We can also use information on uncon-

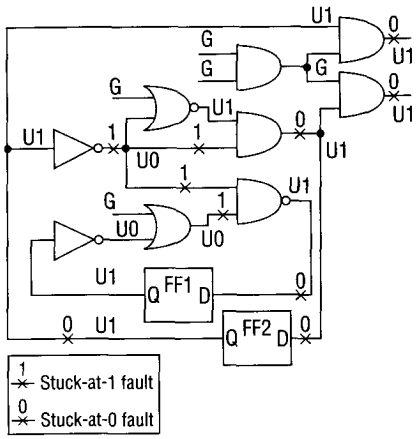


Figure 7. *Unexcitable faults of the example circuit.*

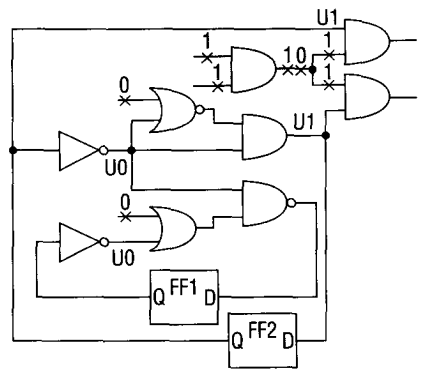


Figure 8. *Unpropagatable faults of the example circuit: UPF1s and UPF2s (shaded area).*

trollable lines in the fault-free circuit to identify this type of untestable fault. Let G be a logic gate with noncontrolling input value α, where α equals 1 for an AND gate and 0 for an OR gate. If we can propagate a fault effect through gate G, then we should be able to justify the input lines of G, except those through which the fault effect will propagate, to have the value α. So if we cannot justify any one of these lines to have the value α, we cannot propagate the fault effect through gate G. From this we derive two rules to identify UPFs.

- Rule 4—Let line l be an input line of gate G that has the noncontrolling input value α. If line l is α-uncontrollable, then the stuck-at faults on all the input lines except line l of G are UPFs.
- Rule 5—Let line l_f be the line where the fault is and let P be the set of all paths that we can propagate the fault effect through from line l_f. If each path in P has at least one gate that blocks propagation of the fault effect through that path, the faults on line l_f are UPFs.

Rule 4 identifies type-1 unpropagatable faults (UPF1s), and Rule 5 identifies type-2 unpropagatable faults (UPF2s). For the example circuit, after deleting the UEFs from the total faults list, we obtain four UPF1s and four UPF2s (Figure 8).

Undrivable faults. In addition to the fault-free circuit characteristics, we also use faulty-circuit characteristics to identify another type of untestable faults. We use the stuck-at-1 fault on line m of the example circuit to explain this idea. With this fault present, we apply a procedure (similar to that for the fault-free circuit) to derive the faulty-circuit characteristics and obtain final results.

As shown in Figure 5, to detect this fault, we must propagate the fault effect to line n and have faulty value 1 on this line. This is because line n is on the only path to the primary output for this fault effect. However, $CH(n)$ equals U, which means we cannot set the faulty value of line n to 1. Hence, we know that this fault cannot be detected and is an untestable fault. We call this an undrivable fault (UDF) since the fault effect itself blocked propagation of this fault. Rule 6 identifies this type of fault.

- Rule 6—After injecting a fault and propagating characteristics, we identify the injected fault as an UDF if the fault effect cannot go through all the paths from the faulty site to primary outputs.

A UDF differs from both a UEF and a UPF. In Figure 3c, $CH(m)$ equals U1, which means that the fault-free value of line m may be set to the value 0 to excite the stuck-at-1 fault on line m. Hence, this fault is not a UEF. In Figure 3c, at the same time, $CH(k)$ equals U0, which means that the fault-free value 1 on line k may be justified while propagating the fault. So this fault is not a UPF either. However, it is an untestable fault (as we have seen in Figure 5) where line n cannot have the faulty value 1 because $CH(n)$ equals U. The fault effect cannot be propagated to line n or any primary output. We categorize these types of faults as UDFs.

Unsensitizable faults. Finally, there is another type of untestable faults that we cannot propagate to any primary outputs because there are no paths between these faults and those outputs. These are unsensitizable faults (USFs). For example, Figure 9 shows the USFs found in the example circuit after we have deleted the other three types of untestable faults. We now easily find unsensitizable faults by checking whether there are paths for the remaining faults to reach primary outputs.

Convergence of symbolic simulation. We model a sequential circuit as a semi-infinite iterative array of combinational circuits C_i in Figure 10. PI_i, S_i, and PO_i represent primary inputs, pseudoprimary inputs, and primary outputs for the circuit in the ith time frame. Assume we start propagating characteristics from C_0 by assigning all characteristics $CH(PI_0)$ to G, all $CH(S_0)$ to U, and all $CH(PI_k)$, where $k > 0$, to G. After propagating characteristics through the iterative array, for each line i_k in C_k, $CH(i_k)$ will belong to one of four cases.

$CH(i_k) = G$. In this case, the characteristic Gs coming from the primary in-

puts of this and/or previous time frames have propagated to line i_k. Once this occurs, all characteristics for line i in subsequent time frames will be G. That is, when a line has a characteristic G in the *k*th time frame, its characteristic will not change to U, U1, or U0 at any *m*th time frame ($m > k$). This means the characteristic of this line is G.

$CH(i_k) = U1$. In this case, the value 1 cannot be set, but we may set the value 0 on line i_k by controlling some primary inputs of this and/or previous time frames. From the first case, line i_k will always be controllable to 0 for C_m ($m > k$). This means that $CH(i_m)$ cannot become U or U0. So $CH(i_m)$ can only be G or U1. For the former, $CH(i)$ converges to G, and for the latter, $CH(i)$ is stable at U1.

$CH(i_k) = U0$. This is similar to case 2. $CH(i)$ will finally stabilize at U0 or G.

$CH(i_k) = U$. From the first three cases, $CH(i_m)$ for $m < k$, cannot be G, U1, or U0; that is, $CH(i_m)$ is U. For $m > k$, $CH(i_m)$ may become G, U1, U0, or U where the first three cases will be stable at G, U1, or U0, and the last one is stable at U.

$CH(i)$ will finally converge to one of these four characteristics.

Experimental results

We incorporated these methods into a sequential-circuit test generation system and identified untestable faults before test generation to save time. Since we removed most untestable faults beforehand, the test generation process spent much less time generating test patterns for the remaining faults.

We ran our experimental results on a Sparc classic workstation using a sequential-circuit test generator based on the line justification strategy.[11] Our sequential-circuit fault simulator used single-event equivalence.[15]

For the example circuit, we used our rules to identify 10 UEFs, four UPF1s, four UPF2s, two USFs, and one UDF. The test generator identified another untestable fault and found two patterns for all eight detectable faults in 0.017 seconds. This was three times faster than when we did not identify untestable faults beforehand. To show the efficiency of this system, we also used it on sequential benchmark circuits.[16,17]

Table 1 shows that the system spent little time calculating controllability, propagating characteristics to obtain the final stable values, and identifying the initializability and invalid states of circuits. The table lists the number of flip-flops which cannot be controlled to 1 or 0. In this column, for circuits s208, s420, s499, s838, and s5378, the identification scheme identified flip-flops using the controllability calculation strategy. If a flip-flop had characteristic U, we counted it twice since it cannot be controlled to either 1 or 0.

The table also lists the number of frames needed to reach a stable condition. We find that the number of frames was rather small and indepen-

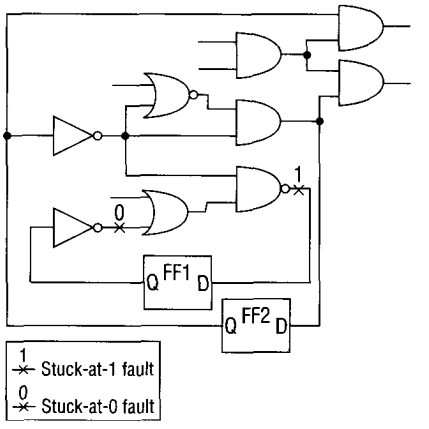


Figure 9. *Unsensitizable faults of the example circuit.*

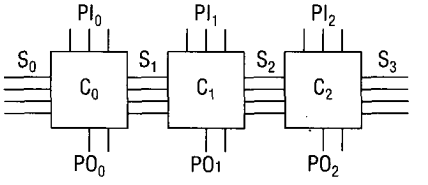


Figure 10. *Sequential circuit modeled as a semi-infinite iterative array of combinational circuits.*

Table 1. *Results of identifying flip-flops with unknown initial states.*

Circuit	No. of flip-flops Total	No. of flip-flops U	No. frames	CPU time (s)
s208	8	3	10	0.02
s420	16	11	18	0.05
s499	22	44	43	0.35
s510	6	12	1	0.00
s641	19	4	5	0.02
s713	19	4	5	0.02
s838	32	27	34	0.12
s953	29	45	3	0.03
s967	29	45	3	0.03
s991	19	38	1	0.03
s1269	37	17	3	0.02
s1512	57	46	4	0.07
s5378	179	37	27	0.68
s9234	228	344	14	5.53
s13207	669	917	26	103.65
s15850	597	654	42	50.13
s38584	1,452	101	39	8.77

Table 2. *Results of untestable fault identification.*

Circuit	Total	No. of faults						Ratio* (%)	CPU time (s)
		UEF	UPF1	UPF2	USF	UDF	Untestable		
s208	215	29	18	11	0	13	71	33.02	0.31
s420	430	99	69	32	0	25	225	52.33	1.75
s499	583	583	0	0	0	0	583	100.00	0.00
s510	564	564	0	0	0	0	564	100.00	0.02
s641	467	27	19	13	0	0	59	12.63	1.21
s713	581	33	20	14	0	4	71	12.22	2.08
s838	857	255	177	96	0	49	577	67.33	9.18
s953	1,079	618	130	239	0	0	987	91.47	1.05
s967	1,066	626	132	228	0	0	986	92.50	0.97
s991	910	593	190	125	0	0	908	99.78	0.17
s1269	1,343	269	48	89	0	2	408	30.38	58.95
s1512	1,357	527	276	487	0	0	1,290	95.06	4.41
s5378	4,603	371	239	326	0	11	947	20.57	731.30
s9234	6,927	5,147	20	202	1,540	0	6,909	99.74	509.55
s13207	9,967	6,923	510	1,120	186	16	8,755	87.84	1,156.73
s15850	11,753	6,126	720	4,818	0	2	11,666	99.26	2,604.34
s38584	36,611	4,199	731	917	0	22	5,869	16.03	211,207.28

* *Ratio of the number of untestable faults to the total faults*

Table 3. *Comparison of other results with this work.*

Circuit	Cheng (Sun 4/260)		Iyer and Abramovici (Sparc 2)		Our method (Sparc classic)	
	No. untestable faults	CPU time (s)	No. untestable faults	CPU time (s)	No. untestable faults	CPU time (s)
s208	*	*	57	1.1	71	0.3
s420	*	*	206	6.0	225	1.8
s713	*	*	32	0.3	71	2.1
s5378	301	1,130	210	25.6	947	732.0
s9234	460	3,686	277	126.1	6,909	515.1
s13207	261	2,793	295	130.4	8,755	1,260.4
s15850	391	5,926	276	311.0	11,666	2,654.5
s38584	*	*	1,332	235.7	5,869	211,207.3

*No data available

dent of the number of flip-flops. The last column shows that the time consumed for each circuit was very little, even for larger circuits.

Table 2 gives the number of each type of untestable fault, as identified using our previously described rules.

In Table 2, the identified untestable faults occupy a large percentage (51.5%) of the total processed faults. CPU time to identify the untestable faults was very short for most circuits. The average time to process one fault was only 0.38 seconds. For larger circuits, the time was still minor compared to that used by the test generator, as we will show later.

Most research[6-10] focuses on finding redundant or expected redundant faults. It is therefore difficult to compare our results to those, since our method finds untestable faults that include redundant and irredundant-but-untestable faults. Nevertheless, we still compare some of our results to those of Cheng,[6] and Iyer and Abramovici[9] in Table 3. Since not all our targeted circuits are the same, we list the circuits in common, the number of

Table 4. Results of test generation with untestable fault identification.

Circuit	No. patterns	No. of faults: Total	Detected	Identified as untestable: Our method	Test generator	Total untestable	Aborted	Fault coverage (%)	Efficiency (%)
s208	179	215	137	71	7	78	0	63.72	100.00
s420	151	430	179	225	26	251	0	41.63	100.00
s499	0	583	0	583	0	583	0	0.00	100.00
s510	0	564	0	564	0	564	0	0.00	100.00
s641	338	467	404	59	3	62	1	86.51	99.79
s713	365	581	476	71	34	105	0	81.93	100.00
s838	216	857	254	577	26	603	0	29.64	100.00
s953	20	1,079	89	987	3	990	0	8.25	100.00
s967	22	1,066	76	986	4	990	0	7.13	100.00
s991	1	910	2	908	0	908	0	0.22	100.00
s1269	58	1,343	240	408	684	1,092	11	17.87	99.18
s1512	16	1,357	66	1,290	1	1,291	0	4.86	100.00
s5378	1,255	4,603	3,090	947	148	1,095	418	67.13	90.92
s9234	5	6,927	18	6,909	0	6,909	0	0.26	100.00
s13207	311	9,967	654	8,755	160	8,915	398	6.56	96.01
s15850	12	11,753	85	11,666	2	11,668	0	0.72	100.00
s38584	5,561	36,611	8,893	5,869	1,693	7,562	20,156	24.29	44.95

identified untestable faults, and CPU time. For these circuits, Table 3 shows that our method used a moderate amount of time to identify many more untestable faults.

Table 4 shows results for test generation and fault simulation. During test generation, we used information on uncontrollable lines and invalid states obtained in the previous processing to improve test generator performance. In the table, efficiency = (detected + total untestable faults) / total faults × 100 Those faults that the test generator failed to identify as untestable and did not generate test patterns for are aborted faults.

Table 5 shows the CPU time required for test generation and fault simulation. Total time spent for untestable fault identification is the sum of the times in Tables 1 and 2. For circuits in which most faults are untestable, we see that this process occupied the major part of the total test

Table 5. CPU time in seconds for untestable fault identification and test generation.

Circuit	Identification	Test generation	Fault simulation
s208	0.33	0.67	0.95
s420	1.80	2.27	1.65
s499	0.35	0.40	0.00
s510	0.02	0.07	0.00
s641	1.23	13.26	2.12
s713	2.10	46.75	2.62
s838	9.30	10.83	12.92
s953	1.08	1.30	1.23
s967	1.00	1.30	1.92
s991	0.20	0.78	0.05
s1269	58.97	5,858	10.13
s1512	4.48	7.31	1.32
s5378	732	201,076	204
s9234	515	522	2
s13207	1,260	28,415	748
s15850	2,654	4,045	22
s38584	211,216	430,121	27,806

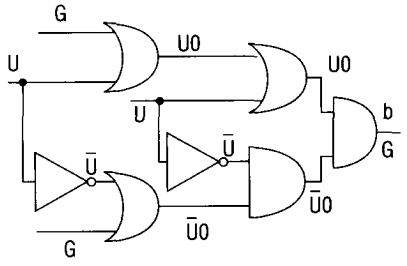


Figure 11. *Example circuit for which a 9-value set of characteristics fails to identify a 1-uncontrollable line b.*

generation time. For these circuits, identifying untestable faults prevented the test generator from wasting time finding test patterns for untestable faults.

For circuits with a few untestable faults, identification occupied a small part of the total test generation time, but the process identified almost all the untestable faults within each circuit. This means that our untestable-fault identification process identified most of the untestable faults in a very short time.

The test generator achieved 100% or over 90% efficiency for most circuits except s38584. We know of no other results for test generation on s38584, however. Comparing the numbers of generated patterns and detected faults with other studies,[11-13] we find that our method achieved higher efficiencies and fault coverages with fewer patterns in moderately less time.

In defining circuit line characteristics, we did not consider the inversion of characteristic U. Thus, some combinationally redundant lines may be incorrectly identified. For example, in Figure 1, if the signal to the input line comes from a flip-flop, line a will have characteristic U. However, the correct characteristic should be U1, since the line is redundant at 0.

To eliminate this error, we used a 9-value set of characteristics. They are U, $\bar{U}$, U1, $\bar{U1}$, U0, $\bar{U0}$, S1, S0, and G, where S1 and S0 mean that a line sticks to 1 and 0. This set of characteristics is more rigorous than the 4-value set, yet it yields results that may not identify some uncontrollable lines. For example, in the circuit of Figure 11, we cannot justify the value 1 on line b. That is, line b is a 1-uncontrollable line but the 9-value set of characteristics cannot identify it. Hence, the identification scheme will detect fewer untestable faults.

In our experiment, we used a 9-value set of characteristics to identify untestable faults for benchmark circuits. Only two circuits, s991 and s13207, had fewer untestable faults (by 5%) using the 9-value set. For s991, the 9-value set identified 863 untestable faults compared to 908 identified with the 4-value set. For s13207, those numbers were 8,491 versus 8,775. Actually, our test generator has verified that, for these two circuits, all the untestable faults identified by the 4-value set but not the 9-value set were real untestable faults. This means that the 4-value set gives correct results for most practical circuits. Even if an error does occur, it occupies only a very small percentage of the total untestable faults.

INFORMATION ON characteristics helps identify uncontrollable lines in sequential circuits and is thus useful in applications such as circuit optimization to eliminate uncontrollable lines. We can also use such information to select flip-flops for partial scan and increase circuit testability. Such an approach is attractive since computing characteristics is simple and takes much less time than other methods, such as structure analysis, testability analysis, and test-generation-based methods. We are currently pursuing such research. D&T

Design and test in Taiwan

With the largest PC production capacity in the world and the world's fourth largest sales volume in ICs (according to Dataquest), Taiwan is becoming a center of intense design and test development. This is evident at an annual VLSI/CAD workshop held in Taiwan, where researchers presented over 70 papers last year. Papers showcased major research results from approximately 10 universities and colleges in Taiwan, as well as government and private research institutes. Approximately 20 papers were on testing.

Heavy design work goes on in Taiwanese industry, with over 50 design houses and 10 fabrication factories capable of designing 16-Mbyte DRAMs and 486-compatible CPUs. However, most testing research activities occur at institutions such as Chiao Tung, Tsing Hwa, Taiwan, Cheng Kung, Central, and Chung Cheng Universities, and Chung Hua Polytechnic College. Altogether, approximately 10 professors and over 50 graduate students engage in design and test research. Their research areas cover almost every aspect of testing, including high-level, gate-level, and delay testing; fault simulation; design and synthesis for testability; BIST; I_{DDQ}, PLA, and memory testing; design verification; defect analysis; yield prediction; and test management. Every year, researchers in Taiwan publish more than 40 papers in these areas.

In addition to the activities in academic circles, some development work also occurs in government-sponsored organizations and private companies. For example, a software house, Syntest, located in Hsin Chu Scientific Industry Park, dedicates itself to developing software tools and consultation services in design and test.

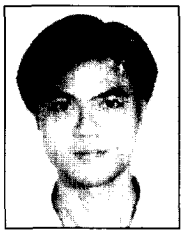
With an increasing number of professionals concerned with design and test, Taiwan is an excellent choice to host the 1996 Asian Test Symposium.

Acknowledgments

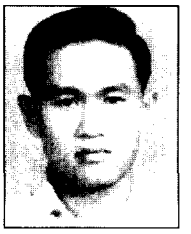
We thank S.M. Reddy, V.D. Agrawal, and the reviewers for their helpful comments.

References

1. M. Harihara and P.R. Menon, "Identification of Undetectable Faults in Combinational Circuits," *Proc. Int'l Conf. Computer Design*, IEEE Computer Society Press, Los Alamitos, Calif., 1989, pp. 290-293.
2. I.M. Ratiu, A. Sangiovanni-Vincentelli, and D.O. Pederson, "VICTOR: A Fast VLSI Testability Analysis Program," *Proc. Int'l Test Conf.*, IEEE CS Press, 1982, pp. 397-401.
3. M. Abramovici, D.T. Miller, and R.K. Roy, "Dynamic Redundancy Identification in Automatic Test Generation," *Proc. Int'l Conf. Computer-Aided Design*, IEEE CS Press, 1989, pp. 466-469.
4. S. Kajihara, H. Shiba, and K. Kinoshita, "Removal of Redundancy in Logic Circuits Under Classification of Undetectable Faults," *Proc. Int'l Symp. Fault-Tolerant Computing*, IEEE CS Press, 1992, pp. 263-270.
5. M. Abramovici and M.A. Iyer, "One-Pass Redundancy Identification and Removal," *Proc. Int'l Test Conf.*, CS Press, 1992, pp. 807-815.
6. K.-T. Cheng, "On Removing Redundancy in Sequential Circuits," *Proc. 28th ACM/IEEE Design Automation Conf.*, IEEE CS Press, 1991, pp. 164-169.
7. J. Moondanos and J.A. Abraham, "Sequential Redundancy Identification Using Verification Techniques," *Proc. Int'l Test Conf.*, IEEE CS Press, 1992, pp. 197-205.
8. V.D. Agrawal and S.T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits," *Proc. European Test Conf.*, IEEE CS Press, 1993, pp. 249-253.
9. M.A. Iyer and M. Abramovici, "Sequentially Untestable Faults Identified Without Search ("Simple Implications Beat Exhaustive Search!")," *Proc. Int'l Test Conf.*, IEEE CS Press, 1994, pp. 259-266.
10. I. Pomeranz and S.M. Reddy, "The Multiple Observation Time Test Strategy," *IEEE Trans. Computers*, Vol. 41, No. 5, May 1992, pp. 627-637.
11. W.-T. Cheng, "The BACK Algorithm for Sequential Test Generation," *Proc. Int'l Conf. Computer Design*, IEEE CS Press, 1988, pp. 66-69.
12. W.-T. Cheng and S. Davidson, "Sequential Circuit Test Generator (STG) Benchmark Results," *Proc. Int'l Symp. Circuits and Systems*, IEEE, Piscataway, N.J., 1989, pp. 1939-1941.
13. T. Niermann and J.H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. European Design Automation Conf.*, EDAC Association, Washington, D.C., 1991, pp. 214-218.
14. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, W.H. Freeman and Company, New York, 1990.
15. C.P. Wu, C.L. Lee, and W.Z. Shen, "SEES-IM—A Fast Synchronous Sequential Circuit Fault Simulator with Single Event Equivalence," *Proc. European Design Automation Conf.*, CS Press, 1992, pp. 446-449.
16. F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Int'l Symp. Circuits and Systems*, IEEE, 1989, pp. 1929-1934.
17. F. Brglez, *ACM/SIGDA Benchmarks Electronic Newsletter DAC93 Edition*, Assoc. for Computing Machinery, New York, Vol. 23, No. 2, June 1993, pp. 39-54.



Hsing-Chung Liang is a PhD student at National Chiao Tung University, engaging in research on VLSI testing and design for testability. Liang holds BS and MS degrees in electronics engineering from National Chiao Tung University. He is a student member of the IEEE and the IEEE Circuits and Systems Society.



Chung Len Lee is a professor in the Department of Electronics Engineering, National Chiao Tung University, where his teaching and research interests focus on integrated circuits and testing. He received his BS from the National Taiwan University and MS and PhD degrees from Carnegie Mellon University. Presently, he serves on the editorial board of the *Journal of Electronic Testing: Theory and Applications*, and is a member of the IEEE Asian Test Technology Committee. He is a senior member of the IEEE Circuits and Systems Society and the IEEE Computer Society.

Jwu E. Chen is an associate professor in the Department of Electrical Engineering, Chung-Hua Polytechnic Institute, Taiwan. His research interests include multiple-valued logic, VLSI testing, synthesis for testability, reliable computing, yield analysis, and test management. Chen received BS, MS, and PhD degrees in electronics engineering from the National Chiao Tung University, Taiwan. He is a member of the IEEE, the Computer Society, AAAS, and NYAS.

Direct questions concerning this article to Chung Len Lee, Department of Electronics Engineering, National Chiao Tung University, 1001 Ta-Hsueh Rd., Hsin-Chu 300, Taiwan, Republic of China; cllee@cc.nctu.edu.tw.