-- --

Surprises in Sequential Redundancy Identification

Mahesh A. Iyer†, **David E. Long***, and **Miron Abramovici** *

† *Synopsys, Inc., Mountain View, CA 94043, USA.*

* *AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.*

Abstract

This paper addresses some misconceptions about redundancy in synchronous sequential circuits. We provide examples to illustrate the differences between untestability and redundancy and discuss existing techniques to identify sequential redundancy. We show that some of these methods are based on incorrect theoretical results. Specifically, we show that untestable faults in balanced pipeline circuits are not necessarily redundant, and that a constant function (i.e. a signal that is always 0 *or* 1 *after initialization) does not always indicate a redundancy. We also show that adding a global reset mechanism or retiming synchronous circuitry may introduce redundancies.*

1. Introduction

Unintentional logical redundancies may be introduced in a circuit without the knowledge of the designer. Identifying these redundancies may provide useful feedback for designers, helping them in locating design errors or finding ways to simplify the circuit. Redundancy identification (RID) can be very useful in synthesis, since every redundant fault defines a region that can be eliminated from the circuit[1] [2].

Automatic test generation (ATG) has often been used to identify redundancy. While this is correct for a combinational circuit, where an untestable fault is always caused by a redundancy, an untestable fault in a sequential circuit is not necessarily redundant[3]. The sequential test generation problem is very complex. Identifying sequential redundancy using ATG has the added complexity of verifying whether an untestable fault is indeed redundant. In general, any practical state-of-the-art sequential test generator cannot distinguish between untestable and redundant faults. Moreover, a practical sequential test generator cannot even be trusted to reliably identify untestable faults. It is well known that because of the pessimism involved in the use of the unknown logic value, a test generator may not find tests for some testable faults[1]. While in ATG this pessimism may only lead to fewer faults being detected, it has serious implications when identifying redundancy, because removing the region associated with an irredundant fault results in a circuit whose behavior is different from that of the original circuit.

In recent years, several methods to identify (and remove) sequential redundancies have been proposed in the literature[2] [4] [5] [6] [7] [8] [9] [10] [11] [12]. In this paper we first present some classes of sequentially untestable but irredundant faults. Next, we discuss the existing methods for sequential RID and show with examples that some of them are in fact based on incorrect theoretical results. We also briefly review the recently proposed notion of c-cycle redundancy[11] [12] and discuss its relation with existing RID methods.

2. Untestability and Redundancy in Sequential Circuits

We use the following definitions from[13]:

Definition 1: A fault f is said to be *testable* if there exists an input sequence I such that for every pair of initial states S and S^f of the fault-free and faulty circuit, respectively, the response $Z(I,S)$ of the fault-free circuit to the input sequence I is different from the response $Z^f(I,S^f)$ of the faulty circuit (at some time on some output).

Definition 2: A fault is *untestable* if it is not testable.

Definition 3: A fault f is *partially testable* if there exists an initial state S^f of the faulty circuit and an input sequence I such that for every fault-free initial state S, the response of the fault-free circuit to I starting from S, $Z(I,S)$, is different from the response of the faulty circuit starting from S^f, $Z^f(I,S^f)$. (This definition differs slightly from that in[13], in that it includes testable faults. This definition makes a testable fault a particular case of a partially testable fault; i.e., a fault is testable if it is partially testable for all initial states of the faulty circuit.)

Definition 4: A fault is *redundant* if it is not partially testable. (A partially testable fault is an irredundant fault.)

Figure 1 illustrates the relations between testable, untestable, and redundant faults.

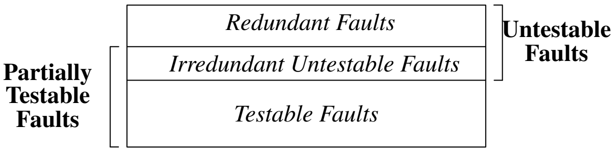


Figure 1. The structure of the fault universe

2.1 Test Strategies

The operation of a sequential circuit typically begins by applying an initialization sequence. When generating tests, we may restrict ourselves to using only a specified initialization sequence (the one intended for use with the circuit), or we may have the freedom to choose an arbitrary sequence. Pomeranz and Reddy[13] define two *test strategies* based on this distinction. Under the *unrestricted test strategy*, tests and partial tests may consist of any sequence of input vectors. Definitions 1-4 as given assume the unrestricted test strategy. Under the *restricted test strategy*, any tests or partial tests must begin with a specified initialization sequence. With this strategy, definitions 1-4 can be modified accordingly. Note that although the restricted test strategy is designed to work with a specific initialization sequence, it imposes no restrictions on the time of observation for all the tests. Faults that are redundant under the unrestricted test strategy are also redundant under the restricted test strategy. The following example illustrates a shortcoming of the restricted and unrestricted test strategies. This example is provided to motivate an alternative definition of redundancy that is introduced in a later section.

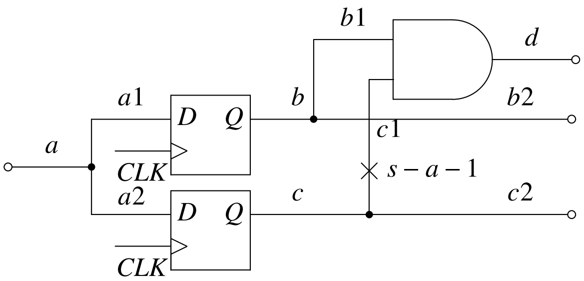


Figure 2. Example of an irredundant fault in the unrestricted and restricted test strategies

Example 1: Consider the circuit in Figure 2 and the fault $c1$ *s-a*-1. This fault is untestable because the requirements to detect it ($c = 0$ and $b = 1$) lead to conflicting assignments on stem a. Independent of its initial state upon power-up, the fault-free circuit can never produce the response $\{d, b2, c2\} = \{1\ 1\ 0\}$. However, if the faulty circuit powers up in state $\{b, c\} = \{1\ 0\}$, this output response will be observed. Hence fault $c1$ *s-a*-1 is partially testable and thus irredundant under the unrestricted test strategy. Now note that any sequence of at least one vector is an initialization sequence for both the fault-free and the faulty circuits. Consider picking some such sequence and using the restricted test strategy. No matter which sequence we choose, the above initial response of the faulty circuit is always possible, and hence the fault is not redundant under the restricted test strategy either. However, the circuit in Figure 2 is intuitively redundant, since the same signal a is fed twice into the same gate d through different flip-flops. □

Note that the irredundant sequentially untestable fault of this example does not affect the initialization of the circuit; i.e. both fault-free and faulty circuits can be initialized with the same sequence. This is contrary to the wide belief that irredundant sequentially untestable faults always affect the initialization of the circuit.

2.2 Irredundant Sequentially Untestable Faults

One class of sequentially untestable but irredundant faults consists of faults that prevent initialization (FPIs)[3]. A method to identify untestable FPIs was introduced in[14]. We now present an example of an untestable FPI that is irredundant.

Example 2: Consider the circuit in Figure 3 and the fault a *s-a*-0. The state transition graphs (STGs) for the fault-free and faulty circuits and the state assignment are shown in Figure 4. An initialization sequence of the fault-free circuit consists of two vectors: $(1, 1)$. In the fault-free circuit, state A is an illegal state. (A legal state is one which is reachable after initialization. An illegal state is a state which is not legal.) Any test sequence to activate a *s-a*-0 must drive a 1 onto line a, independent of the initial states of the fault-free circuit. This is possible only if state A (00) can be reached independent of the initial state of the fault-free circuit. Since in the fault-free circuit state A is illegal, the fault is untestable. However, the input sequence $(1, 1, 0)$ is a partial test for this fault. Independent of its initial state, the output response of the fault-free circuit to this input sequence can never be $(0, 0, 0)$. However, if the faulty circuit started with state A, its response to the input sequence $(1, 1, 0)$ is $(0, 0, 0)$. Hence the fault is irredundant. It is an FPI because, as shown in Figure 4(b), no initialization sequence exists for the faulty circuit. □

Cheng conjectured that partial detection may be possible when the fault-free and the faulty circuits are initialized by different sequences[2] [5]. Now we present a real circuit example to confirm that this conjecture is indeed true and show that the untestable faults which exhibit this behavior form a different class of untestable but irredundant faults. These are the partially testable faults that modify initialization (FMIs).

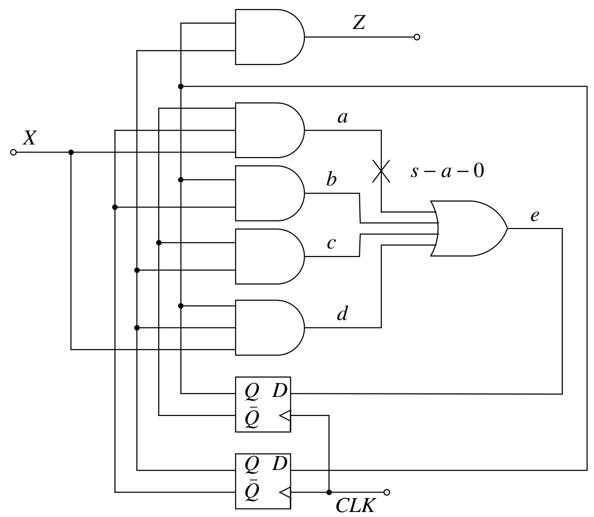
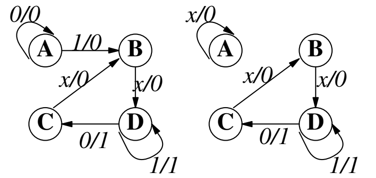


Figure 3. Example circuit with an untestable fault that prevents initialization

(a) Fault-free STG (b) Faulty STG (c) State Assignment

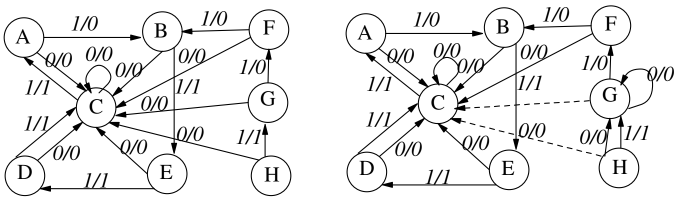


State	Q1 Q2
A	00
B	10
C	01
D	11

Figure 4. State transition graphs for fault-free and faulty circuit of Figure 3

Example 3: The fault-free STG shown in Figure 5 has 5 legal states, $\{A, B, C, D, E\}$ and 3 illegal states, $\{F, G, H\}$. C is the reset state. A detailed gate-level implementation of this circuit can be found in[12]. Let a be the only primary input of this circuit. A single vector, $a = 0$, initializes the fault-free circuit to state C, with output $z = 0$ (see the fault-free STG in Figure 5).

In the gate-level implementation of this circuit[12], a fault is shown that modifies the STG as illustrated in Figure 5(b). After power-up, application of the input sequence $(0, 1)$ brings the fault-free circuit to state A and yields the output sequence $(0, 1)$. However, if the faulty circuit is started from state H, application of the same input sequence brings it to state F and yields the output sequence $(0, 0)$. Thus, the fault is partially testable and can be detected if the faulty circuit starts from state H. It can be verified that the fault is untestable. However, the fault is an irredundant FMI because a different initialization sequence $(0, 1, 0)$ is required for the faulty circuit. □



(a) Fault-free STG **(b) Faulty STG**

Figure 5. The effect of an untestable FMI on the state transition graph

3. Incorrect Solutions to Identify Sequential Redundancy

3.1 The Method of Cheng [4] [5]

Cheng identified special classes of untestable faults that are guaranteed to be redundant. He showed that in a balanced pipeline circuit, every untestable fault is redundant, and that in a general sequential circuit, unactivatable and unpropagatable faults are redundant [4] [5]. (A balanced pipeline circuit is a feedback-free circuit such that, for any signal s in the circuit, every path from any primary input to s has the same number of flip-flops.) He devised special procedures to identify these subsets of redundant faults.

The following example shows that an unactivatable fault is not necessarily redundant. (Activating a fault l *s-a-v* means driving the value $\bar{v}$ onto the faulty line l.)

Example 4: Consider again the circuit and fault in Figure 3. Fault a *s-a-*0 can be activated only by reaching state A, independent of the initial states of both circuits. As discussed in Example 2, this is impossible because state A is illegal. Hence this fault is an unactivatable fault and a always assumes the value 0 after initialization. (This is often referred to as a constant function.) However, as shown in Example 2 and Figure 4, the fault is partially testable and irredundant. Cheng's method [4] [5] will classify this fault as redundant and remove its region from the circuit. But this changes the behavior of the circuit, since the new circuit is uninitializable. □

Next we show that an untestable fault in a balanced pipeline circuit is not necessarily redundant.

Example 5: Consider the balanced pipeline circuit in Figure 6 and the fault $b2$ *s-a-*1. The fault is untestable because its detection requires $b = 0$ and $c = 1$, which causes a conflict on stem a. However, the fault is irredundant under either test strategy. The initialization sequence of the fault-free circuit consists of a single input vector $a = 0$ or $a = 1$; this vector also initializes the output flip-flop g to 0. (Note that a conventional three-valued simulator will not "understand" this functional initialization of flip-flop g. However, the definition of redundancy deals with what the circuit does and not with what a simulator may incorrectly report.) Independent of its initial state, the response of the fault-free circuit to this input vector is $g = 0$. However, if the faulty circuit started with state $\{b, c\} = \{0\ 1\}$, the output response to the input vector $a = 0$ (or $a = 1$) is $g = 1$. Hence $b2$ *s*-*a*-1 is partially testable and irredundant.

While the fault-free circuit is (functionally) initialized by a single vector, note that the faulty circuit requires at least two vectors for initialization. Thus this fault is also an irredundant FMI. □

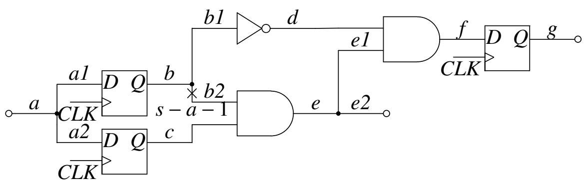


Figure 6. Balanced pipeline circuit with an untestable and irredundant fault

3.2 The Method of Entrena and Cheng[10]

The sequential optimization method of Entrena and Cheng[10] identifies a redundant fault when the mandatory assignments required to detect it cannot be consistently justified. Their analysis is performed only in the fault-free circuit and they claim a fault as redundant either if the mandatory assignments to detect it end up with conflicts or if the state justification procedure starts looping over previously unjustified states. We will now show that both these claims are incorrect. Pomeranz and Reddy have conjectured that the first claim is incorrect and have provided an example to show that the second claim is incorrect [9]. However, their example is not a true counterexample to the second claim in [10], because the mandatory assignment $y1 = 1$ (in their example) is required only in the faulty circuit.

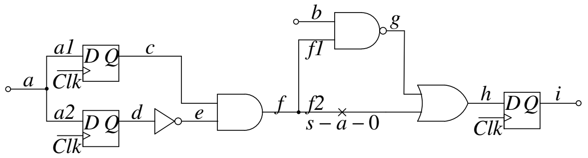


Figure 7. Circuit for Example 7.

Example 6: Consider the circuit in Figure 7 and the fault $f2$ *s*-*a*-0. The fault is untestable because the requirements in the fault-free circuit to detect it lead to a conflict on stem a. Hence the method of Entrena and Cheng [10] will claim this fault as redundant. However, $f2$ *s*-*a*-0 is irredundant under either test strategy for the following reason. An initialization vector of the fault-free circuit is $\{a, b\} = \{0\ 1\}$, which generates the response $i = 1$. (Note that a conventional three-valued simulator using this initialization vector will not initialize flip-flop i.) For the initial state $\{c, d\} = \{1\ 0\}$, the response of the faulty circuit to the same input vector is $i = 0$. Hence $f2$ *s*-*a*-0 is partially testable. □

Example 7: Consider again the circuit in Figure 3. Justification of the mandatory assignments to detect this fault requires justifying the state A (00) in both circuits. As shown by the fault-free STG in Figure 4, state A in the fault-free circuit can only be justified by reaching state A. Clearly the fault is untestable. The method of Entrena and Cheng [10] will claim it as redundant because the state justification procedure in the fault-free circuit starts looping over the previously unjustified state A. However, as shown in Example 2, the fault is partially testable and irredundant. The faulty circuit is uninitializable. □

4. Limitations of Other Solutions to Identify Sequential Redundancy

4.1 The Method of Cheng[2] [5]

Cheng addressed the problem of identifying and removing sequential redundancies without assuming a global reset mechanism. He proposed a procedure to identify a subset of sequential redundancies (called feedback-free sequential redundancies). The method relies on converting a sequential circuit into a feedback-free model and uses a modified sequential test generator to identify sequentially untestable and redundant faults[2] [5]. However, the procedure used to verify whether an untestable fault is indeed redundant is highly complex and is not applicable to large circuits.

4.2 The Assumption of a Fault-free Global Reset Mechanism

Cho *et al.* proposed a sequential RID technique based on implicit state enumeration[7]. The method relies on the simplifying assumption that the circuit has a fault-free global reset mechanism. The algorithm uses an implicit representation of the STG and determines states that are not reachable from the reset state. This information is used by an ATG algorithm to identify sequential redundancies. The approach uses binary decision diagrams (BDDs) as a platform to deal with the reachability information of sequential circuits. Because of the limitations of BDDs, this procedure cannot handle some large circuits.

The methods of[7] [15] assume a fault-free global reset mechanism. Using this assumption, all untestable faults are considered redundant. But the following example shows that this assumption alone does not imply that every untestable fault is redundant.

Example 8: Consider again the circuit and fault a s-a-0 in Figure 3. As shown in Examples 2 and 4, this fault is untestable but irredundant. Now let us examine what happens when a fault-free global reset PI R is added to this circuit and connected to both flip-flops as preset. The STGs of both circuits are now modified such that state D (11) becomes a reset state that can be reached from any state by applying $R = 1$. State A still remains illegal in both circuits and the fault is untestable. Since the global reset mechanism is assumed to be fault-free, the faulty circuit can also be initialized. However, if we analyze the circuit by treating R as a regular PI that can be arbitrarily set to a 0 or 1, we can easily see that the test sequence $(X, R) = (\{1\,0\}, \{1\,0\}, \{0\,0\})$ is a partial test for a s-a-0. Hence the fault is still irredundant under Definition 4. □

This example shows that the assumption of starting not only the normal operation of the circuit with the reset, but every test sequence as well, must be made explicit. This is an important point, because, in general, a circuit in test mode does not have to follow the rules of the normal mode of operation. The results of[7] with respect to RID will be correct only if it is also assumed that every test sequence will *always* start by activating the global reset. In the example above, the partial test for a s-a-0 violates this assumption by not activating R, and no partial test exists that starts with $R = 1$.

Adding a global reset mechanism to a circuit improves its controllability. Since the circuit with the global reset is more controllable, a fault redundant in this circuit is also redundant in the original circuit without the global reset. Example 8 illustrates that imposing the additional constraint that the circuit's operation always starts by activating the global reset can in fact create a redundancy (a s-a-0 is irredundant in the original circuit, but becomes redundant in the circuit with global reset). Thus the method of[7] or any other method that identifies sequential redundancies by assuming a fault-free global reset mechanism (which is always used to start the operation of the circuit), cannot be used to identify redundancies in a circuit without a global reset mechanism, such as data-path circuits.

4.3 The Methods of Pomeranz and Reddy[8] [9]

Pomeranz and Reddy have proposed a method to identify sequential redundancy, based on combinational ATG for multiple faults, which is not practical for large circuits[8]. Their method is an extension of the multi-fault theorem for sequential untestability that was presented in[16] [17]. More recently, they proposed a method to determine whether a sequentially untestable fault prevents the initialization of the circuit[9]. They propose that faults which do not exhibit this behavior can be considered as redundant and the circuit simplified accordingly. The method relies on verifying that the faulty machine has an initialization sequence and requires knowledge of state transition information; thus this method does not seem practical for large circuits. More importantly, their method assumes that the initialization sequence of the circuit can be altered. From a designer's viewpoint this assumption may be unacceptable, since the reset sequence is considered part of the behavior of the circuit. We illustrate this with an example.

Example 9: Consider again the STGs in Figure 5. A detailed gate-level implementation of this sequential circuit and a fault is shown in[12]. The fault was shown to be a partially testable FMI in Example 3. The method of Pomeranz and Reddy[9] will consider this fault as redundant and remove its region from the circuit. The original (fault-free) circuit required only one initialization vector ($a = 0$). This vector will not initialize the new simplified circuit. In fact, a longer initialization sequence of three vectors, $a = (0, 1, 0)$, is required for the new circuit. □

The other limitation of their method is that it is not compositional; i.e. if we are analyzing a subcircuit of a larger circuit, then it is not safe to remove a fault in a subcircuit without first analyzing the full circuit to see whether the whole system will still have a reset sequence. (A property of a subcircuit is compositional if that property still holds when the subcircuit is embedded in any larger circuit.) This is because during initialization some inputs to the subcircuit will come from the larger circuit and some outputs from the subcircuit will be observed by the larger circuit. Since the initialization sequence of the subcircuit may have been altered by using the redundancy removal method of[9], the larger circuit may not be able to provide the new initialization sequence of the subcircuit. Thus, checking that the subcircuit still has an initialization sequence is not sufficient.

4.4 Sequential RID Using Verification

Formal verification techniques were used for sequential RID by Moondanos and Abraham in[6]. Their method relies on verifying that the fault-free and the faulty circuit (for a redundant fault) have equivalent state tables. This definition of sequential redundancy is rather restrictive, since circuit equivalence is not necessary for redundancy. Though their work is of theoretical importance, it is not practical for large circuits, because it relies on building the STGs for the fault-free and the faulty circuits for each modeled fault.

5. Retiming and Redundancy

Circuit retiming[18] consists of changing the position of flip-flops in the circuit without changing the behavior of the circuit. It has been shown that retiming preserves testability[19] [20] [21]. Thus an untestable fault in a retimed circuit will also be untestable in the original circuit. El-Maleh *et al.*[21] claim that retiming cannot introduce new redundancies; the following is a counterexample to this claim.

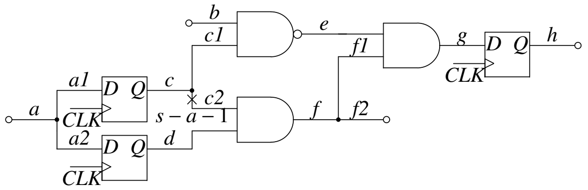


Figure 8. Circuit to illustrate c-cycle redundancy

Example 10: Consider the circuit in Figure 8 and the fault $c2$ s-a-1. The fault is untestable because the requirements to detect it lead to conflicting assignments on stem a. An initialization vector for the fault-free circuit is $\{a, b\} = \{0\ 1\}$, which produces the response $h = 0$. (Note that a three-valued simulator will not "understand" this functional initialization.) However, for the initial state $\{c, d\} = \{0\ 1\}$, the response of the faulty circuit to the same input vector is $h = 1$. Hence the fault is partially testable and irredundant in the unrestricted test strategy and the restricted test strategy that uses the initialization vector $\{a, b\} = \{0\ 1\}$.

Note that the untestability of this fault is caused by the sequential reconvergence of stem a at gate f. However, forward or backward retiming can be used to convert this sequential reconvergence into a combinational one. The fault $c2$ s-a-1 then becomes combinationally redundant in the retimed circuit. □

Dey and Chakradhar[22] observed that retiming can be used to convert sequential reconvergences to combinational reconvergences, and that in the process sequentially redundant faults become combinationally redundant. Example 10 shows that retiming can also transform some irredundant sequentially untestable faults into combinationally redundant faults.

6. c-Cycle Redundancy[11] [12]

Consider any arbitrary input sequence I. If a fault is redundant according to Definition 4, then for *every* state S^f of the faulty circuit, there must exist some state S of the fault-free circuit such that the response of the two circuits (starting from S^f and S) to I is identical. Now suppose that we power up the faulty circuit and clock it c times, applying arbitrary inputs. After this procedure, the set of possible states that the faulty circuit could be in is usually a proper subset of the set of all states, i.e., values in some of the flip-flops will become correlated. Suppose that we only require that every S^f in this smaller set have a corresponding state S in the fault-free circuit for which the response to I is identical. Under this condition, we can still use the faulty circuit as a replacement for the fault-free circuit provided that we clock the faulty circuit c times before applying I.

Definition 5: Consider the set of states $\{S_c\}$ reachable after powering up the faulty circuit and applying any sequence of c input vectors. (Note that $\{S_c\}$ shrinks as c increases.) A fault is c-cycle redundant if it is not partially testable under the assumption that the initial states of the faulty circuit are restricted to $\{S_c\}$.

The concept of c-cycle redundancy is a generalization of the conventional notion of sequential redundancy (Definition 4), which corresponds to 0-cycle redundancy. Also, a c-cycle redundant fault is a c'-cycle redundant fault for any $c' > c$. A circuit with a c-cycle redundant fault can be simplified by removing the region associated with that fault, while this simplification may not be not possible under Definition 4. A c-cycle redundant fault is different from an FMI, since removing a c-cycle redundant fault only requires the application of c *arbitrary* input vectors before the existing initialization sequence, while removing an FMI[9] requires the computation of a new initialization sequence.

Another advantage of c-cycle redundancy is that it is compositional. Hence, removing a c-cycle redundant fault in a subcircuit can be done without examining the larger circuit; we just have to clock the entire system c times before applying its usual initialization sequence.

Example 11: Consider again the circuit in Figure 8 and the fault $c2$ s-a-1. Example 10 showed that the fault is partially testable and irredundant in the unrestricted test strategy and the restricted test strategy that uses the initialization vector $\{a, b\} = \{0\ 1\}$. Now consider applying an arbitrary input vector to the faulty circuit before the input vector $\{a, b\} = \{0\ 1\}$. The application of this arbitrary vector ensures that the FFs c and d have the same value, and makes it impossible to distinguish the faulty circuit from the fault-free circuit. Thus, this fault is a 1-cycle redundant fault. □

It can be verified that $c1$ s-a-1 in Figure 2 and $b2$ s-a-1 in Figure 6 are also 1-cycle redundant. A polynomial-time fault-independent algorithm to identify c-cycle redundancies is described in[11] [12].

--

--

7. Conclusions

This paper has revealed many surprises encountered when trying to solve the sequential RID problem. Since sequentially untestable faults are often mislabeled as redundant faults, we provided examples to illustrate the differences between the two. Existing solutions for sequential RID were analyzed and we showed that some of these solutions are based on incorrect theoretical results. Specifically, we showed that untestable faults in balanced pipeline circuits, untestable faults in circuits with fault-free global reset mechanisms, and unactivatable faults are not necessarily redundant. We also showed that analyzing the mandatory assignments that lead to conflicts do not necessarily identify redundancy. We pointed out that adding a (fault-free) global reset mechanism to a circuit or retiming synchronous circuitry may create redundancies. We also presented the limitations of some other methods. Finally, we reviewed the recently introduced concept of *c*-cycle redundancy and examined its relationship with sequential RID methods.

Acknowledgement

We thank AT&T Bell Laboratories for supporting this research. The support provided by S. Davidson, A. E. Dunlop, and S. Wu made this work possible and is gratefully acknowledged.

REFERENCES

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, “Digital Systems Testing and Testable Design,” IEEE Press, 1994.

[2] K. T. Cheng, “On Removing Redundancy in Sequential Circuits,” *Proc. 28th. Design Automation Conf.*, pp. 164-169, June 1991.

[3] M. Abramovici and M. A. Breuer, “On Redundancy and Fault Detection in Sequential Circuits,” *IEEE Trans. on Computers*, vol. C-28, pp. 864-865, Nov. 1979.

[4] K. T. Cheng, “An ATPG-Based Approach to Sequential Logic Optimization,” *Proc. Intn’l. Conf. on CAD*, pp. 372-375, 1991.

[5] K. T. Cheng, “Redundancy Removal for Sequential Circuits Without Reset States,” *IEEE Trans. on CAD*, vol. 12, no. 1, pp. 13-24, Jan. 1993.

[6] J. Moondanos and J. A. Abraham, “Sequential Redundancy Identification using Verification Techniques,” *Proc. Intn’l. Test Conf.*, pp. 197-205, Sept. 1992.

[7] H. Cho, G. D. Hachtel and F. Somenzi, “Redundancy Identification/Removal and Test Generation for Sequential Circuits Using Implicit State Enumeration,” *IEEE Trans. on CAD*, vol. 12, no. 7, pp. 935-945, July 1993.

[8] I. Pomeranz and S. M. Reddy, “On Identifying Undetectable and Redundant Faults in Synchronous Sequential Circuits,” *12th. IEEE VLSI Test Symp.*, pp. 8-14, April 1994.

[9] I. Pomeranz and S. M. Reddy, “On Achieving Complete Testability of Synchronous Sequential Circuits with Synchronizing Sequences,” *Proc. Intn’l. Test Conf.*, pp. 1007-1016, Oct. 1994.

[10] L. Entrena and K. T. Cheng, “Sequential Logic Optimization By Redundancy Addition and Removal,” *Proc. Intn’l. Conf. on CAD*, pp. 310-315, Nov. 1993.

[11] M. A. Iyer, D. E. Long, and M. Abramovici, “Identifying Sequential Redundancies Without Search,” Technical Report No. IIT-ECE-TR-017-0495, *ECE Department, Illinois Institute of Technology,* Chicago, IL 60616, April 1995.

[12] M. A. Iyer, “On Redundancy and Untestability in Sequential Circuits,” *Ph.D. Thesis, ECE Department, Illinois Institute of Technology*, Chicago, IL - 60616, July 1995.

[13] I. Pomeranz and S. M. Reddy, “Classification of Faults in Synchronous Sequential Circuits,” *IEEE Trans. on Computers*, vol. 42, pp. 1066-1077, Sept. 1993.

[14] M. Abramovici and P. S. Parikh, “Warning: 100% Fault Coverage May be Misleading!!” *Proc. Intn’l. Test Conf.*, pp. 662-668, Sept. 1992.

[15] S. Devadas, H. K. T. Ma, and A. R. Newton, "Redundancies and Don’t Cares in Sequential Logic Synthesis." *J. Electronic Testing: Theory and Application (JETTA)*, vol. 1-1, pp. 15-30, Feb. 1990.

[16] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits." *Proc. European Test Conf.*, pp. 249-253, April 1993.

[17] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits." *IEEE Trans. on CAD*, vol. 14, no. 9, pp. 1155-1160, April 1993.

[18] C. E. Leiserson and J. B. Saxe, “Retiming Synchronous Circuitry,” *Algorithmica*, vol. 6, pp. 5-35, 1991.

[19] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, “Complexity of Sequential ATPG,” *European Design and Test Conf.*, 1994.

[20] 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,* Dec. 1994.

[21] A. El-Maleh, T. Marchok, J. Rajski, and W. Maly, “On Test Set Preservation of Retimed Circuits,” *Proc. Design Automation Conf.*, pp. 176-182, June 1995.

[22] S. Dey and S. T. Chakradhar, “Retiming Sequential Circuits to Enhance Testability,” *12th. IEEE VLSI Test Symp.*, pp. 28-33, April 1994.