Testability of Sequential Circuits with Multi-Cycle False Paths

Priyank Kalla and Maciej Ciesielski
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA-01003

Designated Contact Author:
Prof. Maciej Ciesielski
Department Of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA-01003-4410
Ph: (413)-545-0401
Fax: (413)-545-1993
Email: CIESIEL@illiac.ecs.umass.edu

Abstract

This paper investigates the relationship between multi-cycle false paths and the testability of sequential circuits. We show that removal of multi-cycle false paths (either by circuit restructuring or by proper state encoding) improves circuit testability, but not as significantly as one would expect. We demonstrate the inability of current structure-based scan register selection techniques to select the minimum possible set of registers for high fault coverage. We then propose a novel and efficient way to exploit the *causes* of multi-cycle false paths to make a judicious choice of registers for maximum possible testability. Our technique is based on the analysis of the circuit behaviour and its state encoding. It is shown that this approach results in the selection of minumum possible scan register set for maximum possible fault coverage.

1 Introduction

Over the years, sequential redundancy identification and removal has been the subject of intensive investigation. Identifying unintentional logical redundancies introduced in the circuit provides useful information to the designers about their nature, cause and effect. This information can be exploited to locate design errors and for circuit simplification. Removal of redundancies can aid in testing of circuits and has been known to improve stuck-at-fault (*s-a-f*) coverage. For a combinational circuit, an untestable fault is always caused by redundancy [10], and hence is called a *redundant fault*. The regions associated with such faults can be removed from the circuit. However, an untestable fault in a sequential circuit is not necessarily redundant [8] (it may affect initialization of the circuit). Thus removing regions corresponding to untestable faults results in circuits with a behaviour different from that of the original intended design. In recent years, several methods to identify (and remove) redundancy have been proposed [16] [17] [18] [19]. Automatic test generation (ATG) has been used to identify redundancy. Despite these advances, sequential test generation still remains a complicated problem. Identifying sequential redundancy using ATG has the added complexity of verifying whether an untestable fault is indeed redundant; which leads to added costs during test generation. The non-controllability of the memory elements (latches, F/F's) is believed to be the culprit. For this reason, various design for testability techniques such as full and partial scan are widely used.

In this paper, we address a special type of sequential redundancy, caused by *sequential (or Multi Cycle) false paths*. We analyze the effects of sequential false paths on testability of sequential circuits with feedback (finite state machines). A sequential false path is a natural extention of a combinational false path. Such a path can be identified by analyzing signal propagation over multiple clock cycles, or time-frames [1]. A false path is characterized by the fact that it cannot be activated by any input vector; hence a signal cannot propagate along that path. Consider a path that starts at a primary input and ends at a primary output and contains one or more state registers. Such a path, by definition, spans over more than one time frame, i.e., an input signal takes more than one clock cycle to reach the output. If such a path is statically unsensitizable over a given number of clock cycles, then the signal cannot propagate along this path. Hence it is considered false. Such a path is referred to as a *multi-cycle false path* or *sequential false path*.

A multi-cycle path can be observed by unfolding the combinational logic component of a sequential circuit over several clock cycles, as shown in Fig. 1 (CL_1, CL_2 are identical copies of the combinational logic component CL in two different time frames). Even if the combinational logic part (CL) of the circuit contains no combinational (single-cycle) false paths, the circuit may possess a multi-cycle false path. This happens, for example, when a path between an input to CL_1 and output of CL_2 is unsensitizable over two clock cycles. Tha same concept can be extended to unsensitizable paths over k-clock cycles.

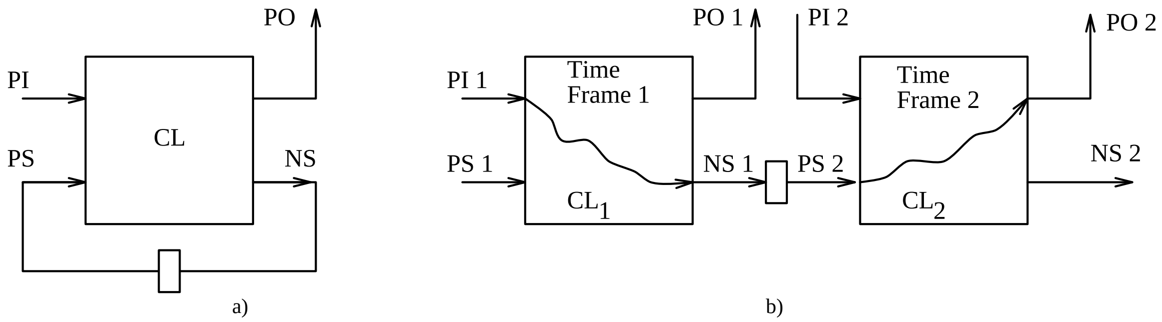


Figure 1: Unrolling a finite state machine over 2 clock cycles

In this paper we investigate the significance of the relationship between multi-cycle false paths and testability of sequential circuits. We analyze the testability of circuits (measured in terms of fault coverage, number of untestable faults, number of test vectors generated, etc.) containing sequential false paths. We present the testability measures for their counterpart circuits implemented so as to be free of false paths. We also investigate the testability measures of the original circuits (ones with false paths) with state-of-the-art scan based techniques. We then propose a novel and efficient way to exploit the properties of sequential false paths to make a judicious choice of scan registers in order to enhance testability using sequential ATPG and partial scan test. We demonstrate that such a technique, when specifically used for circuits with multicycle false paths, gives better results as compared to conventional scan based techniques.

1.1 Previous Work

It has been demonstrated that the presence of multi-cycle false paths in sequential circuits may affect circuit performance, although in a different way than in combinational circuits. Specifically, removal of multi-cycle false paths from a sequential circuit changes the structure of the circuit in such a way that it may lead to performance improvement. *Ashar et. al* [1] have shown that removal of sequential false paths may enable additional retiming, further contributing to cycle time minimization. Their approach to removal of sequential false paths, a gate-level approach, was a straight-forward extension of the KMS algorithm [3].

Hasan et. al. [2] showed that a sequential circuit can be made free of sequential false paths *by construction*, namely by means of state assignment. They showed that multi-cycle false paths are a property of the behaviour of the machine and its state encoding. Subsequently, they proposed a constrained encoding technique that assigns codes to the states such that the synthesized circuit is guaranteed to be false-path-free.

1.2 Contribution of the paper

In this paper, we investigate the relationship of multicycle false paths to *testability* of sequential circuits. While there exist efficient techniques to identify and remove logical redundancies to enhance the testability of combinational circuits, sequential false paths (being a special case of sequential redundancy) and their effect on sequential testability have not been fully researched. We attempt to explain how this type of sequential redundancy can be exploited for the simplification of ATPG-based testing. We also show how to select the scan registers for *partial scan* based testing.

The premise behind removing multi-cycle false paths from circuits to enhance their testability seems to be intuitively sound, as their elimination is a special case of redundancy removal. This paper investigates the validity of this premise. We show that removal of multi-cycle false paths from a circuit does increase testability in most circuits, but to a varying degree. We then present an in-depth analysis of the problem backed up by experimental results. We also suggest a systematic method to *exploit the causes of multi-cycle false paths* so as to judiciously choose registers for *partial scan*. We demonstrate near 100% fault coverage using our procedure, while selecting the minimum possible number of scan flip-flops. We compare our results with conventional scan-register selection techniques and show why such techniques fail to select the best possible set of scan registers for maximum fault coverage and fail to produce minimum number of required test vectors. The procedures of identification and removal of multi-cycle false paths are well understood [3] [2] [1], and are not a subject of this paper.

2 Preliminaries

In this section we review basic terms and definitions related to logic synthesis and testing required to understand analysis of sequential false paths and their relationship to testability. New definitions will be introduced as needed.

A *controlling value* at a gate input is the logic value of the input that independently determines the value at the output of the gate. For example, 0 is a controlling value for an AND gate. The *noncontrolling value* to a gate is the complementary value of the controlling value. For example, 1 is a noncontrolling value for an AND gate.

An *input vector* v is a vector of logic values 0 or 1 at all the primary inputs.

A path is *statically sensitizable* if there exists an input vector which sets all side inputs to the path to non-controlling values. A path is considered to be *statically false* if it is not statically sensitizable.

An Input-to-output (IO) path π_{ij} between input i and output j is said to be *statically sensitizable* in k-clock cycles if there exists an input sequence which will propagate a signal along the path in k-cycles. In this case input i is observable at output j in k clock cycles. We refer to this as a *k-cycle IO observability.*

In testing terminology, the process of finding a nonconflicting assignment of inputs to form a sequence of sensitizing inputs is called *justification*, and the required input vector sequence is called a *sensitizing sequence.* In sequential circuits, the sensitization of multi-cycle paths (referred to as *sequential sensitization*) requires justifying not only primary inputs but also the state registers, which are in general noncontrollable (unless they are scan-registers).

3 Motivation for this research

It is interesting to explore the paths in sequential circuits that are redundant over a given number of clock cycles, *i.e.* paths which are combinationally irredundant but sequentially redundant over k cycles (k being an arbitrary number of clock cycles). The computation performed by k-cycle paths can be conceptually seen as unravelling the combinational part of the circuit over k-time frames, as explained in Section 1. This technique is commonly adopted in test pattern generation.

The circuit shown in Fig. 2, taken from [1], will be used to illustrate the concept of multi-cycle false paths in the context of sequential testability throughout the paper.

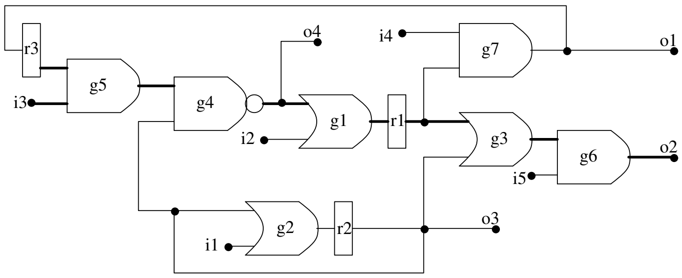


Figure 2: Example circuit with multi-cycle false paths

Analysis of the circuit at the gate level reveals that the two longest two-cycle paths (each of length 5), $r_3, g_5, g_4, g_1, r_1, g_3, g_6, o_2$ and $i_3, g_5, g_4, g_1, r_1, g_3, g_6, o_2$ are false. They are false because in order to sensitize these paths, a 1 is required on the output of register r_2 in the first clock cycle (to sensitize g_4), and a 0 in the second clock cycle (to sensitize g_3). Such an assignment of values to r_2 is not possible because of the feedback at the OR gate g_2. Both of these false paths can be eliminated from the circuit by the method proposed in [1] which is a straight-forward extension of the KMS algorithm [3].

Fig. 3 illustrates the multi-cycle false path removal from the circuit using the algorithm described in [1]. Fig. 3(a) shows the original circuit of Fig. 2 with multi-cycle false paths. Each of the two-cycle false paths is then made fanout-free by path duplication, as shown in Fig. 3(b). The initial value of r_1' is the same as the initial value of r_1 (in this case 0). Both **stuck-at-0** and **stuck-at-1** faults on the fanout of r_3 to g_5' and i_3 to g_5' (labeled as $\times$) are redundant because their effect cannot be propagated beyond a single clock cycle. Therefore, we can replace each of the two wires by constant values, in this case by 1. By propagating these constant values along the false path and simplifying the circuit accordingly gives us the false-path-free circuit of Fig. 3(c). Similar solution (shown in Fig. 3(d)) can be obtained directly by a systematic encoding technique proposed by *Hasan et. al.* [2], which is referred to as the *False-Path-Free* encoding.

A closer analysis of the circuit in the Fig. 3(c) or Fig. 3(d) reveals the following: by removing the 2-cycle false paths, the stuck-at-0 and stuck-at-1 faults at the output of r_3 and at i_3 have been eliminated. This observation, though apparently trivial, highlights the fact that removal of multicycle false paths can help remove redundant faults. This seems to

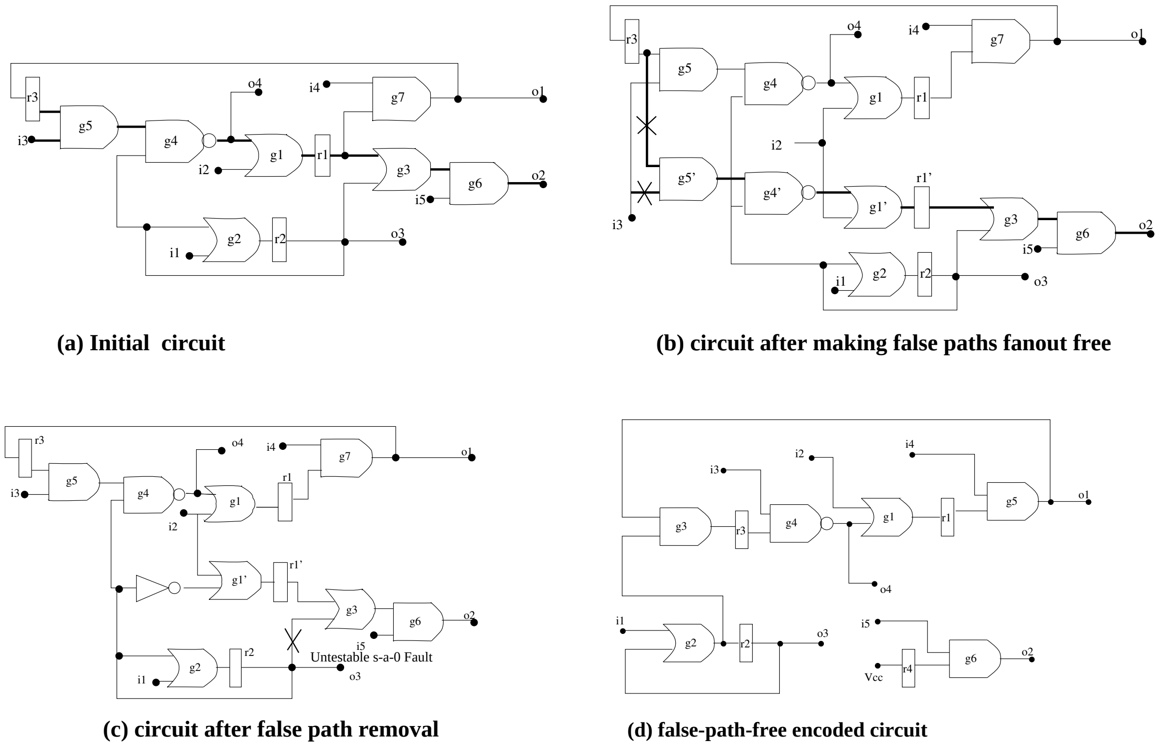


Figure 3: Multi-cycle false path removal

suggest that the number of detected faults in the circuit should increase, and so should the overall fault coverage.

It appears from the above discussion that the notion of k-cycle observability is related to sequential testability. Particularly, if an input is observable in k cycles at some output, then s-a fault at this input is *detectable* at that particular output. Furthermore, any s-a fault along the path to that output is also detectable (provided there are no combinational redundancies). On the other hand, if the circuit has sequential false paths, then some inputs will not be observable at the corresponding outputs, and the s-a faults along these paths will not be detectable. They may or may not be detectable along other paths; but if they are not observable at any other outputs, then these faults are not detectable at all, which leads to a decreased fault coverage. Any attempt of an ATPG to justify the inputs and propagate these faults to the output will fail.

The above observation motivated us to investigate this issue further. We wanted to analyze, in both quantitative and qualitative terms, the effect (if any) that the removal of sequential false paths from circuits has on stuck-at fault coverage. Specifically, we looked into the following issues:

- Does a false-path-free implementation of a circuit really enhance its testability? If so, by what measure? If not, why?
- Removal of multicycle false paths may be expensive, in terms of area and the number of added registers. How does the gain in testability compare against the penalty paid for removing multicycle false paths?
- It would be interesting to compare the increase in testability obtained by removal of sequential false paths, with that obtained by introducing scan registers. Comparisons of the related overheads would also be interesting. How do conventional scan techniques fare in detecting maximum number of stuck at faults for such circuits?
- Exploring the causes and/or characteristics of sequential false paths and their effect on selecting scan registers, that would provide maximum fault coverage, lower CPU time and minimum number of test vectors, is a problem worth investigating. Would it be possible to exploit the characteristics of multi-cycle false paths to select scan registers that provide better results (fewer registers to scan, greater fault coverage, fewer vectors needed) than the current state-of-the-art scan techniques?

We carried out extensive experiments on a set of MCNC benchmarks. Through an in-depth analysis of the results, answers to the above questions were obtained. Multi-cycle false path removal and its effect on testability is explained in later sections. We suggest a new scan register selection technique to be used for the particular class of circuits that contain sequential false paths.

4 The Preliminary Experiment

Hasan et. al. [2], proved that multi-cycle false paths are a property of the behaviour of the circuit and its state encoding. Subsequently, they proposed a systematic method to to predict *a priori* whether an FSM has a potential for an input-to-output path to be false. They introduced the notion of a *potential false path* which is defined as a path that may appear as a false path in a circuit under a certain encoding. Furthermore, they introduced a novel constraint encoding technique that, after detection of a potential false path, assigns codes to the states such that the resulting circuit is false-path-free. This research had culminated in a program, called kMFP, which detects potential false paths over a given number of clock cycles, directly from an STG. It also generates a set of *encoding constraints* which, when satisfied by the encoding, results in a synthesized circuit *guaranteed* to be false path free. The encoding thus obtained is called False Path Free (FPF) encoding. Notice that if these constraints are violated by the state assignment, the implemented circuit will contain multi-cycle false path(s). For our experiments we used the kMFP program, to obtain circuits with a false-path-free (FPF) implementation. We will, from now on, refer to such circuits as FPF encoded circuits.

For our experiments, we analyzed those FSMs which had potential false paths, discarding the rest. These FSMs were encoded in two different ways: A) Using multilevel encoding program JEDI with the *coupled-dominant* algorithm. The synthesized circuits contain multicycle false paths if the assigned codes generated by JEDI do not satisfy the false-path-free encoding constraints. B) Using the FPF encoding, we generated false path free circuits. After synthesizing both circuits, mapping them to a common library, a sequential ATPG tool (HITEC [20]) was used to measure the stuck-at fault coverage, the total number of vectors needed, etc. The flow in Fig. 4 explains our experimental setup.

The standard SIS synthesis is self explanatory. Using JEDI for state encoding and SIS for logic synthesis and technology mapping, a gate level netlist (with sequential false paths) was generated. On the other hand, for the false-path-free synthesis, using kMFP, the constraints on the encoding were generated. These constraints were used by a Dichotomy Input Encoding Tool, DIET [21], to generate the codes for the states. DIET operates on user-defined constraints to generate codes for the states of the FSM. Again, SIS was used for logic synthesis and technology mapping. Finally, HITEC was run on the netlists to obtain their testability measures. Table 1 shows the results of our experiments which were run on all those MCNC benchmarks for which potential false paths were detected.

4.1 Analyzing The Results

As can be seen from the results, the testability of the FPF encoded circuits does increase, but the increase in most cases is not significant. For several FSMs, the removal of sequential false

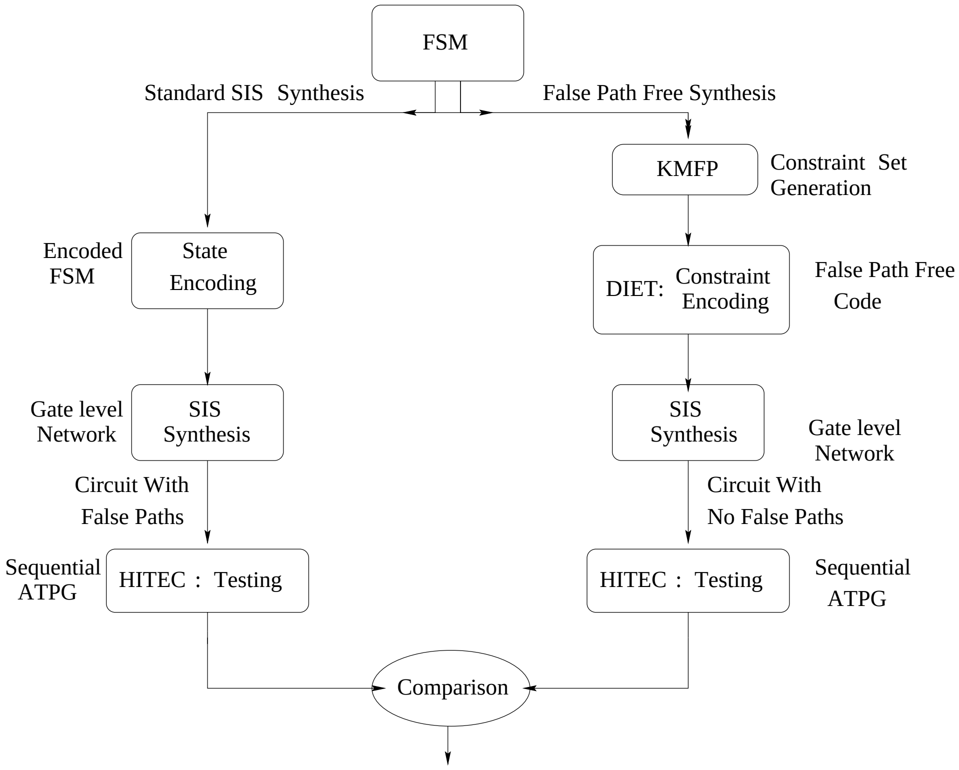


Figure 4: Procedural Flow

paths results in circuits that are not only larger, but contain almost as many undetectable faults. Also, in most cases, the number of test vectors generated for the false-path-free circuits is larger than the number of vectors obtained for the counterpart circuits with false paths. In general, except for a couple of impressive cases, testability of the circuits after removing the multicycle false paths did increase but only marginally. The overhead costs were too high. Not much can be concluded by these results.

Analysis of these results led to the following question:

Q1. *Why did the fault coverage of the circuits not increase to near 100% after removing the sequential redundancies (multi-cycle false paths)?*

Let us consider, again, our example circuit (*false*). Fig. 5 (a) shows the implementation of the circuit that was synthesized by SIS using JEDI encoding. Fig. 5 (b) is its false-path-free implementation. Both circuits were mapped to a library of 2-input gates to simplify the analysis. The testability data for this version is provided in Table 2.

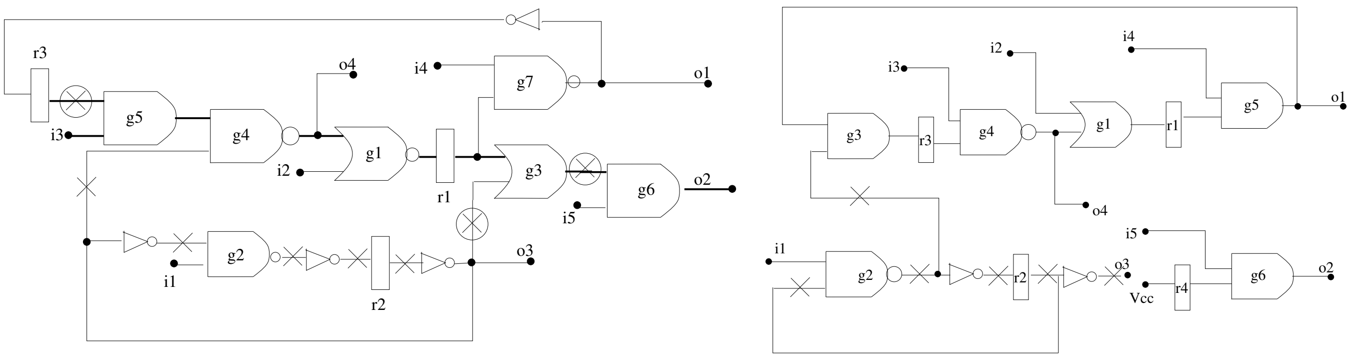
A careful observation of both circuits and the data in Table 2 shows that total number of

Table 1: Experimental results: effect of FPF encoding on Testability

Circuit	Standard circuit					FPF encoded circuit				
	Area	# of regs.	Fault cov.	Untstbl faults	# of vect.	Area	# of regs.	Fault cov.	Untstbl faults	# of vect.
bbsse	148	4	5%	273	9	196	6	95%	17	273
donfile	139	5	100%	0	2	79	6	43%	21	21
ex1	336	5	5%	593	6	284	8	15%	61	49
ex6	126	3	98%	4	99	151	5	99%	5	96
false*	38	3	74%	11	13	91	4	87%	20	81
mark1	119	4	96%	8	80	140	5	97%	9	121
mc	56	2	98%	2	61	54	2	99%	1	42
planet	642	6	1%	1078	4	695	9	3%	1258	8
sse	178	4	5%	273	9	175	8	99%	4	182
s386	163	4	10%	255	16	198	6	97%	9	273
s820	409	5	98%	13	492	341	6	98%	14	565

*example shown in the paper

faults in both the circuits is almost the same (44/42) but there are 3 more untestable faults (marked by a ⊗) in the circuit with multi-cycle false path. These faults are not present in the circuit without false paths. They were obviously removed by eliminating the multi-cycle false path. But there are still 6 untestable faults in the circuit without false path. These faults were not affected by the removal of the multi-cycle false path, as they did not lie on the false path. These faults are associated with the *initializing sequence* of the machine and the *unreachable states*. This can be verified by the STG and the encoding of the circuit.

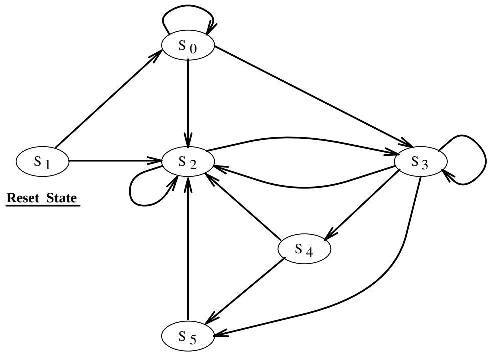


(a) Ckt. with False Path **(b) Ckt. without False Path**

Figure 5: Example circuit after mapping with 2-I/P gates

Table 2: Testability data of the "false" example when mapped on 2-I/P gates

	Standard circuit	FPF encoded circuit
Total # of faults	44	42
# of detected faults	34	36
# Percetage fault coverage	77%	85%
# of Untestable faults	10	6
# of vectors needed	13	15



(a)

State	Code		
	r_1	r_2	r_3
S_0	1	0	0
S_1	0	0	0
S_2	1	1	0
S_3	1	1	1
S_4	0	1	1
S_5	0	1	0

(b)

Figure 6: STG of the example circuit and the corresponding state encoding

Consider the state transition graph shown in Fig. 6, describing the state behaviour of the example circuit. We can see from the STG that once the machine is in one of the states determined by the set $\alpha = (S_2, S_3, S_4, S_5)$ it cannot make a transition to any of the states in the set $\beta = (S_0, S_1)$. Once the machine enters the set α, it remains within α, and there is no path in the STG from α to β. Subsequently, for the encoding in Fig. 6(b) it is not possible to change the value of r_2 from a 1 in the first clock cycle to a 0 in the second clock cycle. Recall that in order to sensitize the false path from i_3 to o_2 (refer to Fig. 3 (a)), we need r_2 to be 1 in the first clock cycle and 0 in the next. But that is not possible for this encoding, and hence the false path. Note that such a behaviour is also the *cause of these untestable s-a faults*, as once r_2 gets a value of 1, it can never change its ouput to 0. Hence the faults associated with the register r_2 (marked $\times$ in Fig. 5), are untestable.

5 Use of Partial Scan

After a rigorous analysis of our example circuit, it becomes clear that removal of multi-cycle false paths eliminates only those *s-a-faults* that lie on the false paths and are redundant.

Removal of multi-cycle false paths does not affect most faults lying on other paths of the circuit. So, for circuits with majority of the untestable faults lying on multi-cycle false paths, circuit restructuring (by means of state encoding or structural removal of multi-cycle false paths) results in the elimination of such faults. This explains the impressive results obtained for a few benchmark circuits shown in Table 1.

It is also clear from the previous discussion that the general lack of *controllability and observability* of the memory elements is the cause of the remaining untestable faults. One way to make the memory elements controllable and observable is the use of scan techniques. It would be interesting to compare the effect of multi-cycle false path removal on testability with that of the use of scan. The *full scan* technique has been developed to simplify the problem of testing a sequential circuit by converting it into a combinational circuit during testing. This enables the application of combinational test generation algorithms, such as the D-algorithm [5], PODEM [6], FAN [7], on such circuits, thus greatly enhancing the testability of the circuits. Although the testing problem is simplified, the area and performance of the circuit are adversely affected as all the memory elements need to be modified to accomodate the complete scan-chain. *Partial scan* on the other hand, provides a trade-off between the ease of testing and the costs associated with scan design. But the key problem is the selection of scan registers. Several papers have been published to define the criteria to guide the selection of the scan memory elements. These techniques are based on i) testability analysis [11], ii) test pattern generation [12], iii) structural analysis [15] and iv) fault-oriented and cost-analysis based approach [14], etc. For our experiments, we used the program called *OPUS* [13] for the selection of scan registers.

Table 3: Testability data of example "false" when r_1 and r_2 are scanned

	OPUS scan (r_1)	Manual Scan (r_2)
Total # of faults	44	44
# of detected faults	37	44
# Percetage fault coverage	84%	100%
# of Untestable faults	7	0
# of vectors needed	13	21

We decided to use partial scan on our example circuit *false*, to find which registers are selected for scan and how this affects testability of the our example circuit that contains multi-cycle false paths. Fig. 7 shows the register r_1 that was selected for scan by OPUS.

Table 3 provides the data obtained by using HITEC on our circuit with r_1 made controllable and observable. Surprisingly, by scanning the register r_1, the fault coverage increased to 84%. Recall from Table 2, the fault coverage of the same circuit rose to 85% by removing the multi-cycle false path also.

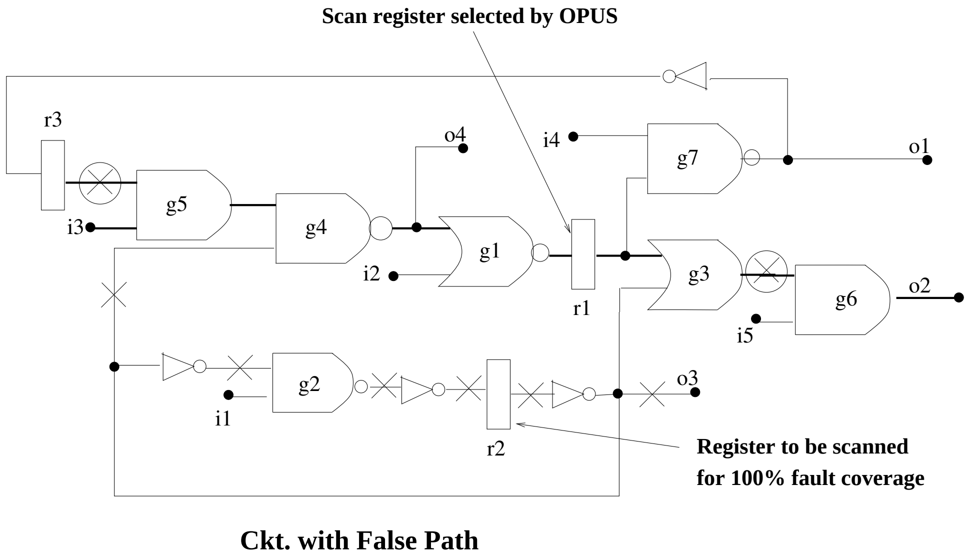


Figure 7: Example circuit depicting the scanned registers

Q2. *Why did the fault coverage of our example circuit not increase to near 100% even after using a scanned register r_1?*

Notice that register r_1 lies on the false path i_3 to o_2. By making that register controllable, it is possible, to test the faults on the false path, that were untestable (marked $\otimes$ in Fig. 7) because of the sequential false path. This was also possible by eliminating the multi-cycle false path. Whereas, the untestable faults caused by register r_2 (marked $\times$ in Fig. 7) were obviously not affected.

We manually selected individually, each memory elements to scan. Scanning the register r_3 gave the same results as scanning r_1 (84%). However, by selecting r_2 for scan, we were able to obtain 100% fault coverage. This was not surprising. This analysis leads us to the following conclusions:

- The cause of all the untestable faults (marked by $\times$ in the Fig. 7) is the lack of controllability and observability of register r_2.

- The register r_2 is also the cause of the multi-cycle false path. Recall that register r_2 cannot obtain a value of 1 in the first clock cycle and a 0 in the next, required for statically sensitizing the multi-cycle path.

Thus by scanning r_2, it becomes possible, for testing purposes, to sensitize the multi-cycle path from i_3 to o_2. Sensitizing this path is made possible by forcing register r_2 to a value 1 in the first clock cycle and to a 0 in the next cycle. Thus the *s-a faults* which were untestable (marked $\otimes$ in Fig. 7) because of the multi-cycle false path, are now made testable. Also, the remaining untestable faults due to the unreachable states (initializing sequence of the machine) can now be tested. Hence the 100% percent fault coverage.

5.1 Optimum selection of scan registers

As shown in the previous section, the conventional scan techniques failed to identify the correct candidate register for partial scan in our example circuit (false). To find out which register should be selected for scan, let us first understand the relationship between the multi-cycle false path and the register r_2 in this example. As discussed in the previous section, the non-controllability of register r_2 makes it impossible to sensitize the path $i_3 \rightarrow o_2$. This non-controllability also introduces the untestable faults. Thus, register r_2 can be considered the *cause* of the sequential false path as well as of the untestable faults. By scanning this register, *i.e.* by making it controllable, we can both sensitize the path that was false (only for testing purposes), and test the faults that were otherwise untestable.

Q3. *Why did the program to select the scan flip-flops, fail to identify r_2 as the best candidate to scan?*

The reason is that the algorithm for scan flip-flop selection used by OPUS is based on *structural analysis* of the sequential circuit [15]. No information about the sequential falsity of input-to-output paths is used for the selection of scan registers. Thus, operating on the network topology it does not exploit any functional information to select best set of scan flip flops.

We shall now show that, for circuits containing multi-cycle false paths, it is possible to select registers, that are the cause of the sequential false paths. The lack of controllability of input and the lack of observability of the output of such registers may also contribute to other sequentially untestable faults. In the next section we shall introduce an algorithm that would exploit such a cause of multi-cycle false paths for the selection of scan registers.

5.2 Suggesting The Algorithm

In the preceding sections, we demonstrated how multi-cycle false paths affect the testability of sequential circuits. We then showed the failure of conventional algorithms for selection of scan registers for maximum fault coverage on our example circuit. We shall now explain, how to select scan registers for sequential circuits with multi-cycle false paths.

Q4. *For circuits with multi-cycle false paths, how can the minimum number of scan registers be selected for maximum fault coverage with least possible number of test vectors?*

To answer this question we will have to refer to the STG of our example circuit and its corresponding state encoding in Fig. 6. Let us analyze the output and next state logic equations of the circuit. Given the state transition conditions and the state encoding one can readily derive the *functional equations* for the circuit. The set of next state and output equations can be derived in terms of the primary inputs $i_i[t]$ and the state variables $r_i[t-1]$. The variables in these equations are also a function of time. The argument [t] in the expressions denotes the discrete time (in clock cycles) and enables us to show the relation between the variables belonging to different time frames. For example, output $o_i[t]$ expressed in terms of state variable $r_j[t-k]$ gives the output expression for o_i in k clock cycles. The following are the *one-cycle* next-state and output equations for the example machine.

$$r_1[t] = i_2[t] + \overline{i_3[t]\ r_2[t-1]\ r_3[t-1]} \quad (1)$$

$$r_2[t] = i_1[t] + r_2[t-1] \quad (2)$$

$$r_3[t] = i_4[t] r_1[t-1] \quad (3)$$

$$o_1[t] = i_4[t] r_1[t-1] \quad (4)$$

$$o_2[t] = i_5[t](r_1[t-1] + r_2[t-1]) \quad (5)$$

$$o_3[t] = r_2[t-1] \quad (6)$$

$$o_4[t] = \overline{i_3[t]\ r_2[t-1]\ r_3[t-1]} \quad (7)$$

We know from the circuit thet there is a false path from i_3 to o_2. Let us consider the output equation of o_2 and its dependance on register variables and inputs. In one clock cycle $o_2[t]$ depends on $i_5[t]$ and on register variables $r_1[t-1]$ and $r_2[t-1]$, which in turn depend on $i_1[t-1], i_2[t-1]$ and $i_3[t-1]$. Let us substitute the variables $r_1[t-1]$ and $r_2[t-1]$ in the expression for $o_2[t]$ by their corresponding one-cycle expressions (1, 2) to obtain a *two-cycle* *expression* for $o_2[t]$,

$$o_2[t] = i_5[t]((i_2[t-1] + \overline{i_3[t-1]\ r_2[t-2]\ r_3[t-2]}) + (i_1[t-1] + r_2[t-2])) \tag{8}$$

$$o_2[t] = i_5[t]((i_2[t-1] + \overline{i_3[t-1]} + \overline{r_2[t-2]} + \overline{r_3[t-2]}) + (i_1[t-1] + r_2[t-2])) \tag{9}$$

Because of the expression $\overline{r_2[t-2]} + r_2[t-2] = 1$ the two-cycle equation for o_2 is reduced to

$$o_2[t] = i_5[t] \tag{10}$$

That is, in two cycles, output o_2 becomes logically independent of i_3. Such a path cannot be sensitized since there is no assignment of side inputs (variables in the expression different than i_3) that will make the output sensitive to i_3; That is why the two-cycle path ($i_3 \rightarrow o_2$) is false.

Going back to our problem of identifying the register to select for partial scan, let us observe the cubes of expression given by 9. The presence of $(r_2[t-2] + \overline{r_2[t-2]})$ was the *cause* of the static unsensitizability of the path from i_3 to o_2. In other words, r_2 masks all the side inputs to the path to o_2. Specifically, *cube* $i_5(r_2 + \overline{r_2})$ covers $i_5 i_3$. Thus, the problem of identifying the best possible scan register for circuits with multi-cycle false path, now reduces to identifying a register cube (related to r_2 in our example) that *covers* an input cube (related to i_3). This can be easily solved by *single cube containment.*

Table 4: Experimental results: Use of Scan

Circuit	OPUS Scan Selected registers	OPUS Scan Fault Coverage	OPUS Scan # of Vectors	Our Proposed Scan Selected registers	Our Proposed Scan Fault Coverage	Our Proposed Scan # of Vectors
bbsse	3	99.66%	69	1	99.66%	131
false	1	85%	10	1	100%	21
mc	1	100%	33	1	100%	21
sse	3	99.66%	69	1	98.7%	142
s386	3	100%	82	1	100%	157

We are in the process of developing this theory into a formal algorithm to be implemented into a computer program. Without the availability of the program it was difficult to analyze the result of applying such an algorithm to circuits with large number of registers. We, however, applied this algorithm manually to the benchmarks shown in Table 4. The results are very promising. For the benchmark circuits *bbsse, sse and s386*, OPUS selected more number of registers (3) than the ones selected by our procedure (just 1). Our procedure selects fewer registers to scan than OPUS, while providing equally high fault coverage. For the benchmark *mc*, OPUS also selects just 1 register, but the register selected by our method is different from that selected by OPUS. Our procedure selects a *better* candidate in this case as it wins over OPUS in the number of test vectors needed. We thus demonstrate the superiority of our method of selecting scan registers over the ones based on structural analysis of the circuits.

6 Conclusions and Future work

This paper analyzed, the effect of removal of multi-cycle false paths on the testability of sequential circuits. The premise behind removing multi-cycle false paths was to enhance the testability of those circuits. As was demonstrated, removal of multi-cycle false paths does affect testability of the circuit in some way or the other. By eliminating the false path, it is possible to test the untestable faults lying on that path. In most cases, the elimination of such a path, however, does not affect the untestable faults in the other parts of the circuit (faults not corresponding to the sequential false paths).

We then analyzed the effect of scan. Conventional scan based approach that operates on the network topology of the circuit, fails to produce the best possible results. That is because they do not use any information provided by the state transition graph (STG) and the state encoding. We then proposed a novel and efficient way to select minimum possible scan registers for maximum fault coverage for circuits that contain multi-cycle false path. Our method exploits the *causes* of the multi-cycle false path and hence produces better results.

This method, however, is applicable to circuits with multi-cycle false paths. Its applicability to circuits without multi-cycle false paths has not been looked into. Hence for circuits without sequential false paths, no claims as yet can be made about this technique. However, it would be interesting to see whether or not such a technique, when combined with other conventional techniques, produce better results in terms of greater fault coverage, minimum scan registers selected, fewer number of test vectors generated, etc.

References

[1] P. Ashar, S. Dey and S. Malik. "Exploiting multi-cycle false paths in the performance optimization of sequential logic circuits", *IEEE Transactions on CAD*, pp 1967-1075, September 1995.

[2] Z. Hasan and M. J. Ciesielski. "Elimination of multi-cycle false paths by state encoding", *In Proceedings of the European Design and Test Conference*, March 1995.

[3] K. Keutzer, S. Malik, and A. Saldhana. "Is redundancy necessary to reduce delay?", *IEEE Transactions on Computer Aided Design*, pp 427-435, April 1991.

[4] M. Abramovici, J. J. Kulikowski, Rabindra K. Roy. "The Best Flip-Flops to Scan", *In Proceedings of The International Test Conference*, 1991.

[5] J. P. Roth, "Diagnosis of automata failures: a calculus and a method", *IBM J. Res. Dev.*, vol. 10, pp 278-291, July 1966.

[6] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits", *IEEE Trans. Computers*, vol. C-30, pp 215-222 March 1981.

[7] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms", *IEEE Transactions on Computers*, vol. C-32, pp1137-1144, Dec 1983.

[8] 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.

[9] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "Complexity of Sequential ATPG", *European Design and Test Conference*, 1994.

[10] M. Abramovici, M. A. Breuer and A. D. Friedman, "Digital Systems Testing and Testable Design", IEEE press, 1994.

[11] E. Trishler, "Incomplete scan path with an automatic test generation methodology", In *Proceedings of the International Test Conference*, pp 153-162, 1980.

[12] V. D. Agarwal, K. T. Cheng, D. D. Johnson and T. Lin, "A complete solution to the partial scan problem", In *Proceedings of the International Test Conference*, pp 44-51, 1987.

[13] V. Chikermane and J. H. Patel, "An optimization based approach to the partial scan design problem", In *Proceedings of the International Test Conference*, pp 377-386, 1990.

[14] V. Chikermane and J. H. Patel, “A fault oriented partial scan design approach”, In *Proceedings of the International Conference on CAD*, pp 400-404, 1991.

[15] D. H. Lee and S. M. Reddy, “On determining scan flip flops in partial scan design”, *Procd. ICCAD*, pp 322-325, 1990.

[16] K. T. Cheng, “On removing redundancy in sequential circuits”, *Proc. 28th Design Automation conf.*, pp 164-169, June 1991.

[17] K. T. Cheng, “Redundancy removal for sequential circuits without reset states”, *IEEE Transactions CAD*, vol. 12, no. 1, pp. 13-24, January 1993.

[18] I. Pomeranz and S. M. Reddy, “On achieveing Complete Testability of Synchronous Sequential Circuits”, *12th IEEE VLSI test Symposium*, pp 8-14, April 1994.

[19] H. CHO, G. D. Hatchel 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.

[20] T. M. Niermann and J. H. Patel, “HITEC: a test generation package for sequential circuits”, In *Proceedings of the European Conf. on Design Automation*, pp 214-218, Feb 1991.

[21] S. Yang and M. J. Ciesielski, “Optimum and suboptimum algorithms for input encoding and its relationship to logic minimization”, *IEEE Trans on Computer Aided Design*, pp 4-12, January 1991.