On Systematic Illegal State Identification for Pseudo-Functional Testing

Feng Yuan and Qiang Xu
CUhk REliable computing laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
Email: {fyuan,qxu}@cse.cuhk.edu.hk

ABSTRACT

The discrepancy between integrated circuits' activities in normal functional mode and that in structural test mode has an increasing adverse impact on the effectiveness of manufacturing test. Pseudo-functional testing tries to resolve this problem by identifying illegal states in functional mode and avoiding them during the test pattern generation process. Existing methods, however, can only extract a small set of illegal states in the system due to various limitations. In this paper, we first show that illegal states in the system are mainly caused by multi-fanout nets in the circuit, and we develop efficient and effective heuristics to identify them. Experimental results on benchmark circuits demonstrate the effectiveness of our proposed systematic solution.

Categories and Subject Descriptors

B.7.3 [**Integrated Circuits**]: Reliability and Testing

General Terms

Reliability, Design.

Keywords

Pseudo-Functional Testing, Illegal States

1. INTRODUCTION

Functional testing was historically used to find manufacturing defects, but its nature of exhaustive testing makes it impractical for reasonable-sized integrated circuits (ICs) [3]. The semiconductor industry hence mainly resorts to structural testing to identify defective chips, wherein test patterns are generated based on circuit structural information and a set of fault models. With the ever increasing transistor-to-pin ratio in IC products, structural test pattern generation has become extremely complex and it is only possible with the help of dedicated design-for-testability (DfT) circuits.

Scan is the mainstream DfT technique in use today, which makes automatic test pattern generation (ATPG) viable for large ICs. Scan-based DfT technique, however, changes the circuit states in test mode, making them possibly different from that in functional mode. Consider a finite state machine encoded with one-hot code, the legal combinations of values in the circuit's storage elements are only those with a single logic '1' and all the others logic '0'. Without taking this functional constraint into consideration, traditional ATPG tools may generate scan patterns that contain multiple logic '1's and hence are illegal in functional mode.

The discrepancy between functional mode and test mode is not a big issue for testing defects that cause slow speed failures, as it mainly affects the timing behavior of the circuit. For ICs fabricated with latest technology, however, at-speed testing is essential to ensure the quality of the shipped IC products, rendering over-testing due to such discrepancy a serious concern for the industry [16, 18, 4]. Recent design evaluations have revealed that at-speed scan patterns were up to 20% slower than any functional pattern [19]. Consequently, some good ICs that would work in application might fail at-speed delay tests, leading to unnecessary *test yield loss* (also known as *test overkill*) [15]. With today's tight profit margins, particularly for chips that go into consumer products, achieving a high manufacturing yield can mark the difference between success and failure, and just a small variation in yield percentage can translate to millions of dollars of revenue change. Therefore, how to reduce test yield loss has become a main concern for the industry.

One way to reduce test overkills is to identify structurally-testable but functionally-untestable delay faults (FU-faults) in the circuit and do not target them during test generation. A significant amount of research work has been conducted in this direction (e.g., [2, 5, 6, 8, 21]). However, FU-fault identification generally has the same complexity as that of sequential ATPG [14], which is exponential in terms of circuit's size. In addition, while test patterns are only generated for those functionally-testable faults in the above methods, it is still possible that they incidentally detect some FU-faults and hence lead to test overkills [14]. Moreover, even for patterns that detect only functionally-testable faults, as we typically let them target as many faults as possible to reduce testing time, they might lead to excessive noises that could not occur in functional mode, again, rendering possible false rejects.

From another perspective, several power-aware test generation methodologies were proposed to reduce test overkills, by reducing switching activities in scan capture mode to ensure the power integrity in manufacturing test [7, 11, 17, 22]. This, however, leads to the concern for under-testing, i.e., if we over-restrict test power, some defective chips containing speed-related defects may pass manufacturing test, leading to *test escapes* [1]. Therefore, the real question is: *How do we make sure that circuits' activities in test mode correlate well with that in functional mode so that we can exercise the worst-case timing of the circuits under test (CUTs) in their functional mode during manufacturing test?*

Lin *et al.* [13] proposed the concept of pseudo-functional testing to tackle the above problem. Instead of identifying FU-faults in the CUT, functionally-unreachable states in the circuit are extracted in [13]. These illegal states are then fed to the ATPG engine to generate *functional-like* patterns. When illegal states are available, the above constrained ATPG process can be conducted quite efficiently as in [9, 14], in which the ATPG tool backtracks immediately when illegal states are reached during test generation.

DAC'09, July 26–31, 2009, San Francisco, California, USA.

Pseudo-functional testing seems to have great potential for resolving the discrepancy problem between structural test mode and functional mode. However, whether we could realize this potential highly relies on whether we could effectively identify as many illegal states as possible. That is, if the extracted illegal states are far from complete, there is still a high possibility that the generated pseudo-functional test patterns are not within the CUT's functionally-reachable space, which invalidates the objective of pseudo-functional testing.

Several approaches were proposed for illegal state identification in the literature, including SAT-based methods [13], implication-based strategies [20, 24], and mining-based techniques [23]. None of the above techniques, however, answers the fundamental question why some states are functionally-unreachable from a structural point of view. They also have their own specific limitations and can only extract a small set of illegal states in the circuit (detailed in Section 2). Consequently, for the success of pseudo-functional testing, more effective techniques are required for illegal state identification.

In this paper, we propose novel solutions to tackle the above problem. The contributions of our work include:

- we show that the illegal states in CUTs are mainly caused by the multi-fanout nets in the circuit.
- we propose novel algorithms that are able to effectively identify much more functionally-unreachable states when compared to state-of-the-art techniques.

The remainder of this paper is organized as follows. Section 2 reviews related prior work and motivates this paper. In Section 3, we study the structural root cause for illegal states. The main flow and the key concept for the proposed illegal state identification scheme are then detailed in Section 4 and Section 5, respectively. Next, Section 6 presents our experimental results on several ISCAS'89 benchmark circuits. Finally, Section 7 concludes this paper.

2. PRELIMINARIES AND MOTIVATION

Illegal state identification has been studied in the context of sequential ATPG in several earlier works [10, 12], wherein they were used to prune the search space for sequential tests. In [10], known illegal states are used to generate larger candidate illegal spaces by eliminating one assignment in the illegal states at a time and trying to justify them. [12], on the other hand, identified invalid states by exploring all valid states through simulation from an unknown initial state. These techniques used in sequential ATPG are not practically viable for today's large ICs due to their extremely high computational complexity.

In [13], Lin *et al.* used a sequential boolean satisfiability (SAT) solver to extract the functional constraints in the system. While theoretically SAT solver is able to find almost all the unreachable states in a circuit, its computational complexity is extremely high and hence cannot be applied to a large circuit. Therefore, [13] proposed to divide the flip-flops (FFs) in the circuit into a number of much smaller groups based on topological analysis and the targeted fault information, each containing few FFs only (e.g., less than 10), and run SAT solver within each group to identify illegal states. Apparently, it is possible that functional constraints exist among different groups and this method cannot identify such kind of illegal states. Moreover, the SAT solver might still abort computation even within the small group of FFs.

Zhang *et al.* [24] proposed an implication-based technique for illegal state identification. The method starts from any gate, say gate A, and finds the implications when its output value is logic '1' and logic '0', respectively. Suppose B and C are internal flip-flops in the circuit, and there exist two implications: $[A(0) \rightarrow B(1)]$ (i.e., $A = 0$ implies $B = 1$) and $[A(1) \rightarrow C(0)]$, we then have $[B(0) \rightarrow A(1)]$ and $[C(1) \rightarrow A(0)]$ according to contrapositive law. Consequently, we can conclude $\{B(0),C(1)\}$ is an illegal state cube. [24] also considered the implications from impossible input-output combinations (e.g., for a 2-input AND gate, when a input is logic '0' while the output is logic '1') in their approach. To keep the computational complexity manageable, [24] implies values based on a single node in one time-frame only. This, however, significantly restricts the number of identified illegal states.

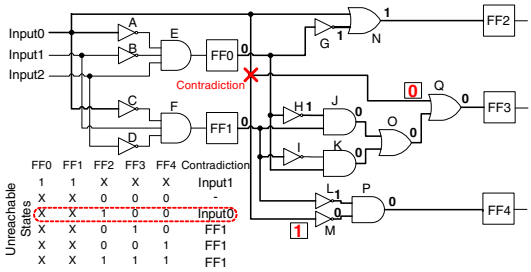

Figure 1: Unreachable State Analysis

Syal *et al.* [20] considered multi-node functional constraints obtained through sequential implications, which cannot be identified within a single time-frame as in [24]. They also advocated to represent illegal states in the form of boolean expressions on arbitrary nets in the circuit instead of values on the state elements. Unlike what the authors claimed, however, using this representation has actually increased the storage overhead for the constraints significantly, because the number of arbitrary nets is much larger than the number of state elements in the circuit and lots of their obtained functional constrains are redundant in nature.

Recently, Wu and Hsiao [23] proposed a mining-based illegal state identification strategy. In this work, the circuit is expanded to multiple time-frames first and then simulated with a number of random patterns. By analyzing the obtained data, some suspicious functional constraints are extracted and they use SAT solver to verify whether they are actually functionally-unreachable. While this dynamic learning method accelerates the search procedure, due to the large amount of data, it can only check pair-wise and three-node relations within small groups of state elements and hence cannot find many illegal states in the system.

To sum up, identifying illegal states using SAT-based [13] or mining-based techniques [23] can be seen as brute-force approaches, which can only be applied to a small circuit at a time. Implication-based methods [20, 24] that consider circuit structural information to tackle this problem look to be more promising. Unfortunately, none of them answers the fundamental question why there are illegal states in the system from the structural point of view. *Finding the root cause of illegal states is extremely important for this problem, as with this information the solution exploration space can be significantly reduced without sacrificing its completeness.* This facilitate pseudo-functional testing to be applicable to real industrial designs. It is the above observation that motivates this work.

3. WHAT IS THE ROOT CAUSE OF ILLEGAL STATES?

Let us examine an example circuit as shown in Fig. 1 to demonstrate our structural root cause analysis for illegal states. This is the gate netlist of a finite state machine that contains six illegal state cubes (see Fig. 1). A closer observation of the circuit shows that except $\{FF2(0),FF3(0),FF4(0)\}$, all the other five illegal state cubes imply logic violation at a multi-fanout net. For example, let us try to justify illegal state $\{FF2(1),FF3(0),FF4(0)\}$. First, we can learn $Input0(0)$ and $O(0)$ through OR gate Q, and $G(1)$ and $FF0(0)$ through OR gate N. We can then derive $J(0)$ and $K(0)$ through OR gate O and $H(1)$ through the inverter H. Next, $FF1(0)$ is justified through AND gate J. Finally, we can infer $Input0(1)$

through AND gate P with $FF1(0)$ and $FF4(0)$, which, however, contradicts to the aforementioned $\{Input0(0)\}$ justified through another fanout going through gate Q. On the other hand, although the illegal state cube $\{FF2(0), FF3(0), FF4(0)\}$ do not imply any logic violation at a multi-fanout net directly, it actually infers another illegal state cube $\{FF0(1), FF1(1)\}$. In other words, this particular illegal state also causes logic violation on a multi-fanout net $Input1$ implicitly, which occurs at another time frame. This example motivates us to consider whether multi-fanout nets are the main root cause of illegal states.

DEFINITION 1. *Consider a circuit that does not contain any multi-fanout nets, denoted as circuit C. FF_i is the i^{th} flip-flop in circuit C. The flip-flop set $\mathcal{F}$ contains all flip-flops that do not belong to any sequential loop. The flip-flop set $\mathcal{L}_j$ contains all flip-flops belonging to the j^{th} sequential loop, and $\mathcal{L}$ ($\mathcal{L} = \bigcup_j \mathcal{L}_j$) is the set of all flip-flops belonging to any sequential loops.*

DEFINITION 2. *$\mathcal{S}(FF_i)$ is defined as the **father-cone set** of FF_i, including all state elements (including both primary inputs and flip-flops) that directly determine the next state of FF_i. **Ancestor-cone set** $\mathcal{A}(FF_i)$ includes all state elements that directly or indirectly determine the state of FF_i in one or more clock cycle(s).*

LEMMA 3. *In circuit C, $\mathcal{S}(FF_k) \bigcap \mathcal{S}(FF_n) = \emptyset$ when $k \neq n$.*

PROOF. If there exist a common state element within $\mathcal{S}(FF_k)$ and $\mathcal{S}(FF_n)$, there must be a multi-fanout net in the circuit, which is conflicted with the definition of circuit C. ∎

LEMMA 4. *The states of any elements in $\mathcal{L}_j$ cannot affect the states of elements outside of this set.*

PROOF. Suppose there is a flip-flop FF_k belonging to the flip-flop set $\mathcal{L}$, whose father-cone set $\mathcal{S}(FF_k)$ includes at least one flip-flop FF_n belonging to $\mathcal{L}_j$. Since all flip-flops in $\mathcal{L}_j$ are connected one by one to form a sequential loop, FF_k also belongs to the father-cone set of another flip-flop FF_q in $\mathcal{L}_j$. Therefore, $\mathcal{S}(FF_k)$ and $\mathcal{S}(FF_q)$ have a common state element, which contradicts Lemma 3. ∎

DEFINITION 5. *The* sequential level *of a flip-flop FF_i in set $\mathcal{L}$ is set to be the maximal sequential level of all state elements in $\mathcal{S}(FF_i)$ plus one, assuming the sequential level of all primary inputs is 0.*

THEOREM 6. *In a circuit C, suppose the maximum sequential level of the elements in $\mathcal{L}$ is n, any state of $\mathcal{L}$ can be reachable within n clock cycles from the primary inputs.*

PROOF. We prove this theorem by mathematical induction principle. Let us start with the case $n = 1$. Apparently, any states on Level 1 flip-flops are reachable within one clock cycle. This is because Level 1 flip-flops are directly determined by primary inputs, and the father-cone sets of any two Level 1 flip-flops are independent. Next, suppose any state for flip-flops with level k is reachable within k clock cycles, where $k \leq n$. In this case, all flip-flops with level $n+1$ are reachable within $n+1$ clock cycles. This is because, Level 1 flip-flops have essentially the same functionality as primary inputs after one clock cycle. In addition, the flip-flops with level 2 to n that directly connected with the flip-flops with level $n+1$ can also be viewed as primary inputs, without multi-fanout nets in the circuit. Therefore, any state for flip-flops with level $n+1$ is reachable. ∎

The above theorem proves that no functionally-unreachable states exists in the *NLF* of *CS*. However, it cannot guarantee the same conclusion holds true for sequential loop structure. Take the circuit shown in Fig. 2 as an example, we observe that if $FF0$ and $FF1$ are initialized as $\{FF0(0), FF1(0)\}$, it would never escape from this state even if we change the states of $FF2 - FF5$, the other

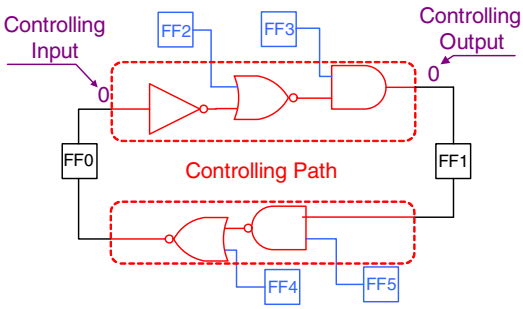

Figure 2: Sequential Loop-Induced Unreachable States

three states for $FF0$ and $FF1$, therefore, are not reachable. Such sequential loop-induced illegal states are very difficult, if not impossible, to be found by any automatic illegal state identification method. This is because: (i). sequential loop expands their effects to different sequential levels in infinite number of time frames; (ii). which states are reachable depends on the initial state of the involved flip-flops.

Fortunately, the illegal states caused by sequential loop are rare cases, because the conditions to form sequential loop-induced unreachable states are quite stringent: (i). between any connected flip-flops on the loop, there must be a controlling path as shown in Fig. 2, which is a logical path where a controlling value can be directly propagated from the beginning to the end; (ii). all controlling paths should have transitivity, i.e., the output controlling value of a path should be the input controlling value of the next path. If any one of the above two conditions is not true, any states in the sequential loop is functionally-reachable. For example, in Fig. 2, suppose we just change the NOR gate which connects with $FF0$ to an OR gate, the controlling path from $FF1$ to $FF0$ is broken. The value of $FF0$ is not solely dependent on its previous on-loop flip-flop $FF1$. Starting from this flip-flop, we first set its value by assigning $FF4 = 1$ and $FF5 = 1$ such that $FF0$ can be flipped to logic 1 and then controlling path between $FF0$ and $FF1$ will be also broken. Therefore, the state of the sequential loop in next clock cycle can be determined by $FF2 - FF5$, thus can reach any state. Moreover, even for these rare sequential loop-induced unreachable states, their impact on testing is rather limited because the involved state elements typically span a number of clock cycles while we target defects in combinational logic between adjacent cycles in testing.

Because of the above reasons, we can simply ignore the unreachable states caused by sequential loops without damaging the effectiveness of pseudo-functional testing much.

4. ILLEGAL STATE IDENTIFICATION FLOW

As discussed earlier, illegal states would imply logic violations at different branches of the same multi-fanout net, explicitly or implicitly. A first thought to generate illegal states is then to propagate contradictive logic values at different branches of multi-fanouts concurrently, and find out which state cubes can justify this combination. These state cubes can then be deemed as illegal. We initially tried out for this intuitive method and found out it is not a good solution. This is because: (i). we need to avoid the case that two fanout branches with contradictive values propagate through the same logic elements (as we cannot determine its value under such circumstances), which results in incomplete identified illegal states; (ii). as we need to propagate contradictive logic values at the branches of each multi-fanout net pairwisely, for those nets that contain a high number of branches, we need to propagate along each branch multiple times and it is a waste of computational efforts. To resolve the above problems, instead of propagating contradictive values at multi-fanout nets, we determine which state cubes

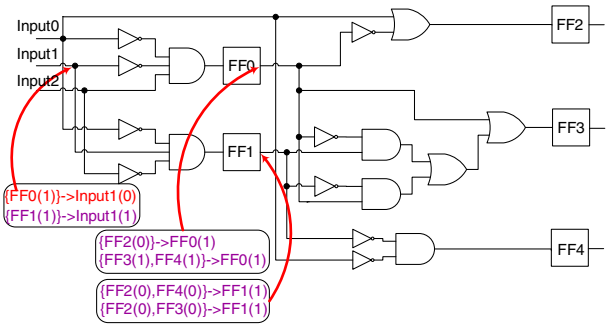

Figure 3: Example Circuit for Illegal State Identification

can justify logic '1'('0') at the multi-fanout nets independently and use this information to obtain illegal states. We use the example circuit shown in Fig. 3 to explain the main flow of the proposed illegal state identification scheme, as depicted in Fig. 4.

Let us define the so-called *justification scheme* at every circuit node in the format of $Cube0 \rightarrow 0$ and $Cube1 \rightarrow 1$, denoting that a state cube $Cube0/Cube1$ justifies logic '0/1' on this node. For example, a justification scheme $FF0(1) \rightarrow Input1(0)$ in Fig. 3 means that $FF0 = 1$ can justify logic '0' at circuit node $Input1$. All such justification schemes are systematically built for each net before unreachable state extraction (detailed in Section 5). According to previous discussion, we always start from a multi-fanout net and try to derive illegal states using contradictory justification schemes. In this example, for the multi-fanout at $Input1$, we have $\{FF0(1)\} \rightarrow Input1(0)$ and $\{FF1(1)\} \rightarrow Input1(1)$. We can therefore conclude that the state cube $\{FF0(1), FF1(1)\}$ is illegal as they justify contradictory values in this fanout. The above procedure needs to be conducted for every multi-fanout nets.

As the illegal state cubes obtained from different paths may contain redundant information (e.g., the set of illegal state cubes shown in Fig. 1 is not the most compact one), we need to remove such redundancy so that all the cubes are minimized and disjoint to each other. This step is conducted by gradually building up a hypergraph for identified illegal state cubes. That is, each flip-flop (FFx) is split into two vertices ($FFx(0)$ and $FFx(1)$) to denote its two possible logic values, and an illegal state cube can be represented as a hyperedge that connects the corresponding vertices. We define the following relationships between hyperedges A and B:

- if A connects to only a subset of vertices of B, we denote this relationship as A *dominates* B. Apparently, the corresponding illegal state cube for hyperedge A contains that of B;
- if both A and B connect to n vertices, and among them $n-1$ vertices are the same and the remaining one corresponds to the same flip-flop with different logic values, we denote this relationship as A *and* B *complement each other*. These two hyperedges can be replaced by a single hyperedge that connects to the $n-1$ common vertices according to our definition.

Every time we add a new hyperedge into the graph, we check the above relationships for its related edges and finally we get a compact set of illegal state cubes without redundancy. For example, for the illegal state cubes shown in Fig. 1, they can be finally compacted into five illegal state cubes: $\{FF0(1), FF1(1)\}$, $\{FF3(0), FF4(0)\}$, $\{FF2(0), FF3(0)\}$, $\{FF2(0), FF3(1), FF4(0)\}$, and $\{FF2(1), FF3(1), FF4(1)\}$, using the above method.

Next, we expand the current illegal state cubes to the next sequential level, again, using the justification scheme information. For the example circuit in Fig. 3, according to structural analysis, two justification schemes at $FF0$ and $FF1$ are detected. We explore

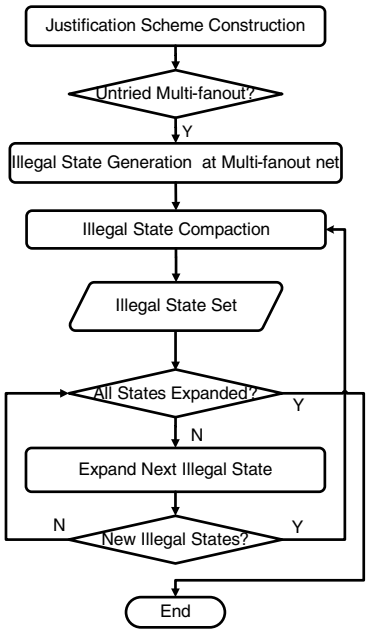

Figure 4: Flowchart for the Proposed Illegal State Identification Scheme

all possible combinations of expanded unreachable cubes, and obtain new cubes, e.g., $\{FF2(0), FF4(0)\}$ and $\{FF2(0), FF3(0)\}$. As shown in our flowchart in Fig. 4, this step is conducted in a stack-like manner. That is, whenever we generate new illegal state cubes through expansion, they will be pushed into the illegal state set for later expansion as well. Compared to prior work that explicitly expand the circuit into a few time frames for illegal state justification, the above procedure is able to implicitly walk through unlimited number of time frames of the circuit efficiently. Finally, the whole procedure for our illegal state justification terminates when there is no unexpanded illegal state.

So far we have discussed how to conduct illegal state extraction, expansion, and compaction, with available justification schemes at every circuit node. In the following section, the key issue in our algorithm, how to construct these justification schemes in an efficient and effective manner, is discussed in detail.

5. JUSTIFICATION SCHEME CONSTRUCTION

We start our justification scheme construction at the input of each flip-flop, which can be obtained directly with the state of the flip-flop. We then propagate these initial justification schemes to every circuit node based on the following propagation rules:

Rule 1: Backward Propagation

A non-controlled value at the output of a logic gate (e.g., logic '1' for a NOR gate) will imply non-controlling value for its inputs of this logic gate. Therefore, a justification scheme that justifies a non-controlled value at the output of a logic gate can be propagated to all inputs of this logic gate to justify the non-controlling value there, referred as *backward propagation*. The backward propagation of justification schemes over a two-input NOR gate is shown in Fig. 5(a) as an example.

Rule 2: Forward Propagation

A justification scheme that justifies the controlling value at any input of a logic gate also justifies the controlled value at its output (e.g., logic '1' for an OR gate). If this scenario occurs, we can propagate the justification scheme from the input to the output directly, referred as *forward propagation*. Similarly, an example for two-input NOR gate is shown in Fig. 5(b).

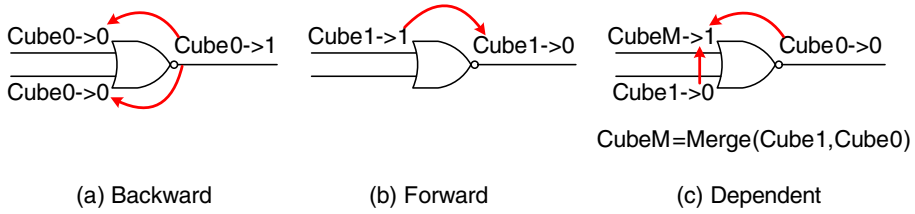

Figure 5: Propagation Rules for Justification Schemes

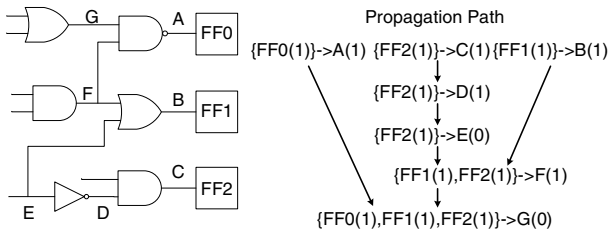

Figure 6: Generation of Sophisticated Justification Schemes

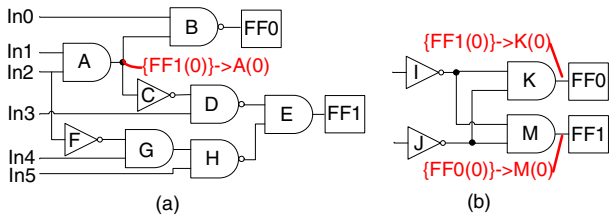

Figure 7: The Impact of Reconvergent Nodes

Rule 3: Dependent Propagation

Generally speaking, to justify a controlling value at a certain input of a logic gate, we need to justify its output to be the controlled value as well as all other inputs to be the non-controlling value. Therefore, dependency exist when propagating such justification schemes and we need to merge state cubes to obtain such justification schemes, referred as *dependent propagation*. Again, an example for NOR gate is shown in Fig. 5(c). Please note that, no justification schemes would be generated if there is any conflict when merging state cubes.

Let us use an example circuit (see Fig. 6) to show how we can build sophisticated justification schemes based on the above propagation rules. After initialization, three schemes $FF0(1) \rightarrow A(1)$, $FF2(1) \rightarrow C(1)$ and $FF1(1) \rightarrow B(1)$ are built. Then, justification scheme $FF2(1) \rightarrow C(1)$ is propagated along path $C-D-E$ and we can obtain justification scheme $FF2(1) \rightarrow E(0)$ at E. Together with $FF1(1) \rightarrow B(1)$, we have $\{FF1(1), FF2(1)\} \rightarrow F(1)$. According to the above and $FF0(1) \rightarrow A(1)$, we can finally obtain $\{FF0(1), FF1(1), FF2(1)\} \rightarrow G(0)$. The last two schemes which is generated based on dependent rule cannot be extracted by implication based technique since such method can only generate justification schemes between single flip-flop and single node.

The above propagation rules, however, are not enough for us to build complete justification schemes. For example, for the example circuit shown in Fig. 7(a), if we would like to propagate justification scheme $FF1(0) \rightarrow E(0)$ backwardly, as logic '0' is a controlled value for AND gate E, based on dependent propagation rule, we need to merge $FF1(0)$ with the cube that justifies $H(1)$ (i.e., $FF1(1)$) to obtain the cube that justifies $D(0)$. Apparently, this merging is not possible and hence we have to stop propagation through gate D. Consequently, we cannot obtain justification scheme $FF1(0) \rightarrow A(0)$ based on our earlier propagation rules. This justification scheme, however, does exist, because $E(0)$ implies $D(0)$ and/or $H(0)$ while both $D(0)$ and $H(0)$ imply $A(0)$. Similarly, the justification schemes shown in Fig. 7(b) cannot be obtained with the above propagation rules. A closer examination of the circuit shown in Fig. 7(a) reveals that the existence of the justification scheme $FF1(0) \rightarrow A(0)$ is due to the fact that gate E is a reconvergent node of multi-fanout net $In2$. That is, both inputs of gate E are affected by $In2$ and this essentially leads to the implication of $A(1) \rightarrow In2(1) \rightarrow E(1)$ and hence $FF1(0) \rightarrow E(0) \rightarrow A(0)$. For the circuit in Fig. 7(b), even though there is no reconvergent node in the traditional sense, considering logic cubes are propagated both forwardly and backwardly, gates K and M can be also deemed as *virtual* reconvergent nodes.

As the occurrence of the above missing justification schemes can be attributed to reconvergent nodes, to resolve this problem, we conduct implication at every circuit node and we record those implications that lead to the same non-controlling value of a logic gate (if not, there is no reconverging effects). For example, by doing so, we can learn $A(1)$ implies the non-controlling logic '1' at both inputs of logic gate E. Hence we have $A(1) \rightarrow E(1)$ and we would record $E(0) \rightarrow A(0)$, which can be used during propagation through gate E to obtain $FF1(0) \rightarrow A(0)$. Based on the same principle, we would record $K(0) \rightarrow M(0)$ and $M(0) \rightarrow K(0)$ for the circuit shown in 7(b), which can then be used to build the justification schemes shown in the figure. It is important to note that, those implications that do not lead to the same non-controlling value of a logic gate would be discarded, as they can be obtained based on our propagation rules.

By conducting the above implication procedure before the justification scheme propagation process, we are able to build complete justification schemes for the circuit.

6. EXPERIMENTAL RESULTS

To evaluate the effectiveness of the proposed solution, we perform experiments on several ISCAS'89 benchmark circuits, comparing against the implication-based algorithm in [24], which is shown to be able to obtain much more illegal states than the SAT-based method in [13]. We do not compare against [20] and [23] because they do not store functional constraints in the format of illegal states (in arbitrary nets instead). Our experiments are conducted on a 2GHz PC with 1GB memory.

In [24], the authors reported the total number of illegal state cubes only. Let us first compare this value obtained using the two algorithms, as shown in Table 1 (see Columns 2 and 7). Apparently, we can find much more illegal states when compared to [24]. It should be also highlighted that our algorithm are generally more effective for those large circuits.

As one 2-bit illegal state cube can be represented as two 3-bit illegal state cubes, this makes comparing the total number of illegal state cubes not quite dependable. As a result, for fair comparison, we re-implement the algorithm in [24] and all illegal state cubes detected by the two algorithms are compacted in the same manner to remove redundancy. In Table 1, we classify all the illegal state cubes according to the number of specified bits. We can observe

Benchmark	[24]					Proposed							
	Total #	# of 2-bit	# of 3-bit	# of 4-bit	Running Time	Total #	# of 2-bit	# of 3-bit	# of 4-bit	# of 5-bit	# of Large	# of Expanded Cubes	Running Time
s208	16	16	0	0	0.00	23	23	0	0	0	0	0	0.01
s444	16	12	4	0	0.00	49	14	20	15	0	0	0	0.00
s953	116	101	15	0	0.05	365	153	133	78	1	0	0	12.32
s1196	35	10	25	0	0.02	138	10	43	42	20	23	0	0.68
s5378	1006	293	711	2	5.23	8516	399	2584	266	2383	2884	681	153.86
s9324	195	47	148	0	2.30	1109	47	168	108	204	582	17	19.65
s13207	2808	1222	1330	256	19.48	82651	1445	4156	14002	27760	34928	60140	151.28
s15850	507	112	392	3	20.32	5568	239	399	777	261	3892	65	211.82
s38417	988	483	496	9	56.30	90983	864	3094	16544	34641	35840	42559	397.58

Table 1: Experimental Results for Illegal State Identification.

that most illegal state cubes detected by [24] are 2-bit or 3-bit cubes (there are no cubes with size more than 4). This is because their rather simple implication scheme can only obtain relatively small-sized functional constraints. The proposed method, on the other hand, is able to generate a large amount of large-sized illegal state cubes (i.e., cube size larger than 3). More importantly, our method also obtains much more 2-bit and 3-bit cubes than [24], which is able to restrict the functional space more effectively. It can be also observed from the table that our illegal state expansion technique is quite effective. For certain benchmark circuits (e.g., s13207), the illegal cubes generated by expansion can reach nearly 80% of the total illegal states.

Finally, we carefully examine the illegal state cubes generated from the two algorithms, and we find out that all the illegal states obtained in [24] are covered in the illegal state cubes obtained using our proposed method. This further proves that multi-fanout nets are the main structural root cause for functionally-unreachable states, and gives us more confidence to realize the potential of pseudo-functional testing using the proposed methodology.

7. CONCLUSION

The discrepancy between ICs' activities in normal functional mode and that in structural test mode has an increasingly adverse impact on the effectiveness of manufacturing test with technology advancement, as reported in several industrial studies (e.g., [4, 18, 19]). Pseudo-functional testing seems to have great potential to resolve this problem, but whether we could realize this potential highly relies on whether we could effectively identify as many illegal states as possible. In this paper, we show that the main structural root cause for illegal states is the multi-fanout nets in the circuit. Based on this observation, we develop efficient and effective algorithms for illegal state identification. Experimental results on ISCAS'89 benchmark circuits demonstrate that the proposed technique is much more effective when compared to state-of-the-art solutions.

8. ACKNOWLEDGEMENTS

This work was supported in part by the General Research Fund CUHK417406, CUHK417807, and CUHK418708 from Hong Kong SAR Research Grants Council, and in part by a grant N_CUHK417/08 from the NSFC/RGC Joint Research Scheme.

9. REFERENCES

[1] Moderator: K. Butler, Organizer: N. Mukherjee. Power-Aware DFT - Do We Really Need it? Panel, International Test Conference, 2008.

[2] D. Brand and V. S. Iyengar. Identification of Redundant Delay Faults. *IEEE Transactions on Computer-Aided Design*, 13(5):553–565, May 1994.

[3] M. Bushnell and V. Agrawal. *Essentials of Electronic Testing*. Kluwer Academic Publishers, 2000.

[4] C. Shi and R. Kapur. How Power Aware Test Improves Reliability and Yield. EE Times, Sept. 15, 2004.

[5] G. Chen, S. M. Reddy, and I. Pomeranz. Procedures for Identifying Untestable and Redundant Transition Faults in Synchronous Sequential Circuits. In *Proceedings International Conference on Computer Design (ICCD)*, pages 36–41, 2003.

[6] K.-T. Cheng and H.-C. Chen. Classification and Identification of Nonrobust Untestable Path Delay Faults. *IEEE Transactions on Computer-Aided Design*, 15(8):845–853, August 1996.

[7] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer. Low Power Scan Shift and Capture in the EDT Environment. In *Proceedings IEEE International Test Conference (ITC)*, paper 13.2, 2008.

[8] K. Heragu, J. H. Patel, and V. D. Agrawal. Fast Identification of Untestable Delay Faults Using Implications. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 642–647, 1997.

[9] M. Konijnenburg, J. van der Linden, and A. van de Goor. Test Pattern Generation with Restrictors. In *Proceedings IEEE International Test Conference (ITC)*, pages 598–605, 1993.

[10] M. Konijnenburg, J. van der Linden, and A. van de Goor. Illegal State Space Identification for Sequential Circuit Test Generation. In *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pages 741–746, 1999.

[11] J. Li, Q. Xu, Y. Hu and X. Li. iFill: An Impact-Oriented X-Filling Method for Shift- and Capture-Power Reduction in At-Speed Scan-Based Testing. In *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pages 1184-1189, 2008.

[12] H.-C. Liang, C. L. Lee, and J. E. Chen. Identifying Invalid States for Sequential Circuit Test Generation. *IEEE Transactions on Computer-Aided Design*, 16(9):1025–1033, Sept. 1997.

[13] Y.-C. Lin, F. Lu, and K. Cheng. Pseudofunctional Testing. *IEEE Transactions on Computer-Aided Design*, 25(8):1535–1546, August 2006.

[14] X. Liu and M. S. Hsiao. A Novel Transition Fault ATPG that Reduces Yield Loss. *IEEE Design & Test of Computers*, 22(6):576–584, Nov.-Dec. 2005.

[15] P. Maxwell, I. Hartanto, and L. Bentz. Comparing Functional and Structural Tests. In *Proceedings IEEE International Test Conference (ITC)*, pages 400–407, 2000.

[16] J. Rearick. Too Much Delay Fault Coverage Is A Bad Thing. In *Proceedings IEEE International Test Conference (ITC)*, pages 624–633, Nov. 2001.

[17] S. Remersaro, X. Lin, S. M. Reddy, I. Pomeranz, and J. Rajski. Scan-Based Tests with Low Switching Activity. *IEEE Design & Test of Computers*, 24(3):268–275, May-June 2007.

[18] J. Saxena, K. Butler, V. Jayaram, and S. Kundu. A Case Study of IR-Drop in Structured At-Speed Testing. In *Proceedings IEEE International Test Conference (ITC)*, 2003.

[19] S. Sde-Paz and E. Salomon. Frequency and Power Correlation between At-Speed Scan and Functional Tests. In *Proceedings IEEE International Test Conference (ITC)*, paper 13.3, 2005.

[20] M. Syal, K. Chandrasekar, V. Vimjam, M. S. Hsiao, Y.-S. Chang, and S. Chakravarty. A Study of Implication Based Pseudo Functional Testing. In *Proceedings IEEE International Test Conference (ITC)*, page paper 24.3, 2006.

[21] M. Syal and M. S. Hsiao. New Techniques for Untestable Fault Identification in Sequential Circuits. *IEEE Transactions on Computer-Aided Design*, 25(6):1117–1131, 2006.

[22] X. Wen, K. Miyase, T. Suzuki, S. Kajihara, Y. Ohsumi, and K. K. Saluja. Critical-Path-Aware X-Filling for Effective IR-Drop Reduction in At-Speed Scan Testing. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 527–532, June 2007.

[23] W. Wu and M. S. Hsiao. Mining Sequential Constraints for Pseudo-Functional Testing. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 19–24, 2007.

[24] Z. Zhang, S. Reddy, and I. Pomeranz. On Generate Pseudo-Functional Delay Fault Tests for Scan Designs. In *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 215–226, 2005.