Formal Verification-Based Redundancy Identification of Transition Faults with Broadside Scan Tests

Hiroshi IWATA, Member, Nanami KATAYAMA, Nonmember, and Ken’ichi YAMAGUCHI, Member

SUMMARY In accordance with Moore’s law, recent design issues include shortening of time-to-market and detection of delay faults. Several studies with respect to formal techniques have examined the first issue. Using the equivalence checking, it is possible to identify whether large circuits are equivalent or not in a practical time frame. With respect to the latter issue, it is difficult to achieve 100% fault efficiency even for transition faults in full scan designs. This study involved proposing a redundant transition fault identification method using equivalence checking. The main concept of the proposed algorithm involved combining the following two known techniques, 1. modeling of a transition fault as a stuck-at fault with temporal expansion and 2. detection of a stuck-at fault by using equivalence checking tools. The experimental results indicated that the proposed redundant identification method using a formal approach achieved 100% fault efficiency for all benchmark circuits in a practical time even if a commercial ATPG tool was unable to achieve 100% fault efficiency for several circuits.

key words: redundancy identification, transition faults, broadside scan test, formal verification and functional equivalence

1. Introduction

According to Moore’s law, an increase in the number of transistors leads to an increase in the scarcity of resources such as power and interconnection bandwidth. This results in a design productivity gap in which the number of available transistors grows faster than the ability to meaningfully design the transistors [1]. Therefore, shortening time-to-market is an important problem.

Given this demand, several studies have suggested formal techniques to shorten time-to-market. Equivalence checking techniques that can process large practical circuits in a practical time given that the correspondence between flip-flops of two circuits is available and the combinational part can be extracted. An application of equivalence checking involves establishing correspondence between a high-level abstract design and a low-level implementation [2].

From the testing viewpoint, the trend involves delay faults such as transition faults associated with high-speed clock networks and nano-scaling. Practically, transition faults are detected with a full scan design that is developed from manufacturing test targeting stuck-at faults. More precise delay test models, such as small delay quality models (SDQM) [3] and path delay fault models [4]–[6], were proposed to treat stacking small delay defects. However, it was not possible to achieve the requisite test quality for large practical circuits in an applicable time frame. Additionally, it was not possible to achieve complete (100%) fault efficiency even with respect to transition faults. A major factor in this situation involves identifying redundant faults. In order to identify a fault is redundant, it is necessary that no test pattern exists from the entire test pattern. Unfortunately, a considerable number of redundant faults appears in gate level designs due to the lack of design optimization.

This study focused on functional equivalence between a fault-free circuit and a faulty circuit with a redundant fault. The characteristics of the stuck-at fault are widely-known topic. In the study, the characteristics were upgraded to transition faults, and a rapid redundant identifying method with commercial equivalence checking tools was proposed. The main advantage of using commercial equivalence checking tools includes the fast identification of functional equivalence given that many internal equivalent points exist in the circuits. With respect to identifying a redundant fault, the fast identification can be available because most combinational gates have the same structure in fault-free and faulty circuits.

The rest of this paper is organized as follows. Experimental studies are introduced in Sect. 2. Preliminaries are presented in Sect. 3. Section 4 introduces our proposed redundant identification method using formal verification tools. Experimental results are presented in Sect. 5. Section 6 concludes the findings of the study.

2. Related Works

The problem of functional equivalence between fault-free and redundant fault injected circuits is widely-known. Thus, several studies proposed a redundant identification method using formal approaches. Moondanos et al.[7] proposed a method using functional equivalence of redundant fault in sequential circuits to identify redundant sequential stuck-at faults. Without using functional equivalence of redundant faults, Abramovici et al.[8] proposed a method to identify redundant stuck-at faults using controllability and observability. Another study [9] proposed a method that could represent a control flow of a design as a finite-state machine while not specifically targeting redundant fault identifica-
In order to alleviate over-testing, several methods that identify redundant transition faults with a full scan design were proposed\cite{10, 11}. The application of a full scan design allows the detection of many sequential redundant faults because transitions to functional unreachable state occur. As a result, over-testing problems exist that increase yield loss by detecting sequential redundant faults.

In this study, a method was proposed to identify redundant transition fault in full scan designs. The proposed method had the following two advantages.

1. Complete redundant identification was achieved. That is, every fault could be identified as redundant or detectable, and the test pattern was also available if the fault was detectable.
2. In order to identify redundancy, equivalence checking tools were also used as ATPG for circuits with many equivalent points.

3. Preliminaries

3.1 Fault Model

In this study, the stuck-at fault model and transition fault model were used as the logical and delay fault models, respectively. A stuck-at fault changes a signal line to a fixed value \( v \), and this is denoted by \( s-a-v \). Conversely, a transition fault causes sufficient delay that exceeds the clock period at a signal line. This delay is denoted either as slow-to-rise \((s-t-r)\) if the delay occurs on the rising edge or slow-to-fall \((s-t-f)\) if the delay occurs on the falling edge. Equivalent faults are collapsible to a representative fault to reduce the large number of the faults in a circuit. A fault (denoted as \( f \)) is termed as redundant if a fault-free circuit (denoted as \( C \)) and the faulty circuit (denoted as \( C_f \) in which \( f \) exists in \( C \)) are functionally equivalent. Thus, the simplification of injecting redundant fault \( f \) is applicable to circuit \( C \).

3.2 Circuit Model

In this study, a full scan designed circuit that consisted of multiplexer-based scan flip-flops and a combinational circuit was considered. All the flip-flops in the circuit were connected in series with single or multiple scan chains. The full scan design was widely accepted as the de-facto standard for design for testability (DFT) since any input pattern was applicable for the combinational circuit, and the output response at the circuit was observable. Therefore, from a testing viewpoint, the full scan sequential circuit could be treated as a combinational circuit. It was difficult for the full scan design to apply two consecutive test patterns. Hence, two models existed and were classified according to the method of applying the second test pattern into broadside and skewedload models. Broadside and skewedload models use the second test pattern as the output response of the combination circuit by applying the first pattern, and the scan-shifted pattern of the first pattern, respectively. The broadside expansion model was discussed as follows. Nevertheless, the skewedload expansion model could also be treated by a slight modification of the temporal expansion method.

A full scan circuit could be temporally expanded by duplicating the combinational part of the circuit and connecting the pseudo primary inputs/outputs corresponding to the test pattern application model. Figure 1(a) shows a full scan sequential circuit that consists of \( n \) primary inputs, \( m \) primary outputs, \( k \) flip-flops, and a combinational part \((C_o)\). The value of every flip-flop was completely controllable and observable by using a scan shift operation. Hence, inputs and outputs of the combinational part were treated as pseudo primary inputs and outputs, respectively. It was assumed that the detection of a slow-to-rise transition fault \((s-t-r)\) on the signal line \( s \) produced a rising transition in which the first pattern corresponded to ‘0’ and second pattern corresponded to ‘1’, and the transition was propagated to any (pseudo) primary output. The first pattern could be set with any value, but the second pattern was determined by the output response of the first pattern if a broadside expansion model was used.

Fig. 1 Circuit model and its temporal expansion models.
Figure 1(b) shows the temporally expanded circuit with a broadside expansion model including the restriction between the first pattern and the second pattern. The circuit consists of two circuits \( C_1 \) and \( C_2 \) that are copies of the combinational part \( C_o \) in the full scan design. In this expanded circuit, the primary inputs of \( C_1 \) and \( C_2 \), and pseudo primary inputs of \( C_1 \) were directly controllable. Additionally, the primary outputs and pseudo primary outputs of \( C_2 \) were directly observable. The pseudo primary inputs of \( C_2 \) were connected to the corresponding pseudo primary outputs of \( C_1 \) to represent the second pattern restriction. It was not necessary to observe the response of \( C_1 \) since the error occurred during the transition in \( C_2 \). In order to represent the rising transition at \( s \) in the expanded circuit, the value ‘0’ was set as \( s_1 \) and value ‘1’ is set as \( s_2 \). However, the slow-to-rise transition fault on \( s \) did not change the value of \( s_2 \) from ‘0’ to ‘1’. That is, it was equivalent to a stuck-at-0 fault. Therefore, the transition fault was represented as the value restriction of \( s_1 \) and the stuck-at-fault on \( s_2 \).

Figure 1(c) shows another version of the expanded circuit. In the above model (Fig. 1(b)), two arbitrary consecutive test patterns were applied to the primary inputs and the response of the second pattern was captured at primary outputs at-speed (that is, the operational speed of the circuit during the test). Hence, it was necessary for the test equipment to work faster than the operational speed of the circuit under test. In contrast, a PI/P0 restricted model assumed that all the patterns and responses were applied and captured at the scan flip-flops. The restriction could be ignored if the flip-flops were connected after all the primary inputs and before all the primary outputs. The assumption was practical for high-speed systems in which the input/output delay was ignorable.

### 3.3 Functional Equivalence between Redundant Fault Injected and Fault-Free Circuits

The transition fault could be treated as a stuck-at fault with a value restriction as discussed in Sect. 3.2. The functional equivalence of redundant stuck-at faults is a known characteristic. In this section, the functional equivalence for redundant transition faults was defined based on a broadside expansion model. First, the relationship between redundant stuck-at fault and its functional equivalence was considered. This relationship was then advanced to the transition fault with the broadside expansion model.

**Definition 1:** For a combinational part \( C_o \) and a stuck-at \( v \) fault on a signal \( s \) in \( C_o \), let \( \alpha \) be the fault and \( C_{\alpha} \) be the faulty circuit. The fault injected circuit \( C_{\alpha} \) is obtained by cutting \( s \) of \( C_o \) and connecting a static value \( v \) to the end of \( s \). Hence, \( \alpha \) is redundant iff \( C_o \) and \( C_{\alpha} \) are functionally equivalent.

The contraposition of Definition 1 is that if \( C_o \) and \( C_{\alpha} \) are not functionally equivalent, there exists a test pattern detecting the stuck-at fault \( \alpha \). The test pattern is obtained by the counter-example calculated with equivalence checking tools. Therefore, equivalence checking tools were treated as ATPG.

As discussed in Sect. 3.2, the transition fault could be treated as a stuck-at fault with a value restriction. In order to represent the value restriction during equivalence checking, a broadside ATPG model was defined by Definitions 2 and 3 that follows. Additionally, the transition fault injected model was defined in Definition 4.

**Definition 2:** For a transition fault \( \alpha, v \) is a restricted value determined by the following condition.

\[
v = \begin{cases} 
0 & \text{if } \alpha \text{ is slow-to rise fault,} \\
1 & \text{if } \alpha \text{ is slow-to fall fault.} 
\end{cases}
\]

**Definition 3:** For a combinational part \( C_o \) and a transition fault on a signal \( s \) in \( C_o \), let \( \alpha \) be the fault, \( v \) be the restricted value, \( C_1 \) and \( C_2 \) be copies of \( C_o \), and \( s_1 \) and \( s_2 \) be the signals mapped from \( s \) in \( C_1 \) and \( C_2 \), respectively. In order to restrict the value of \( s_1 \) to \( v \), as a result of the following two circuit modifications, the resulting circuit is called as the broadside ATPG model, and denoted as \( C^R(v) \).

1. Connect every PO and PPO of \( C_2 \) to an input of restricted value gate (2-input OR gate and AND gate if \( v = 0 \) and \( v = 1 \), respectively)
2. Add a fan-out to \( s_1 \), and connect the fan-out to the other input of every restricted value gate.

Figure 2 shows an example of the broadside ATPG model \( C^R(0) \) for a slow-to-rise fault on \( s \). In order to obtain counter-examples, equivalence checking tools restricted the value of \( s_1 \) to ‘0’ since every output was constant ‘1’ if the value of \( s_1 \) corresponded to ‘1’.

**Definition 4:** For a combinational part \( C_o \) and a transition fault on a signal \( s \) in \( C_o \), let \( \alpha \) be the fault, \( v \) be the restricted...
value, $C_1$ and $C_2$ be copies of $C_o$, and $s_1$ and $s_2$ be the signals mapped from $s$ in $C_1$ and $C_2$, respectively. With respect to the broadside ATPG model $C_R(v)$, stuck-at $v$ fault on $s_2$ injected circuit $C_R(v)$ is called as the transition fault injected model, and denoted as $C_I(v)$.

Figure 3 shows an example of the transition fault injected model $C_I(0)$ (that is, $C_R(0)$ where $\beta$ is $s.a.0$ on $s_2$) for a slow-to rise fault on $s$. The model was obtained by injecting the stuck-at 0 fault to $s_2$ in the broadside ATPG model $C_R(0)$.

Given the above definitions, the functional equivalence of the redundant transition fault is as follows.

**Theorem 1:** Let us consider a transition fault $\alpha$ and the restricted value $v$. $\alpha$ is redundant if the transition fault injected model $C_I(v)$ and the broadside ATPG model $C_R(v)$ are functionally equivalent.

**Proof:** Specifically, given that $C_I(v)$ and $C_R(v)$ are functionally equivalent, this implies that $v$ is set as $s_1$ and $s.a.v$ on $s_2$ is redundant. In this situation, either no transition occurs with respect to $s$ or the transition does not affect the (pseudo) primary output. Therefore, $\alpha$ is redundant.

If $v$ cannot be set as $s_1$, then, the value of $s_2$ is fixed as $\overline{v}$, that is $s.a.\overline{v}$ exists on $s$. Therefore, $C_I(v)$ and $C_R(v)$ are functionally equivalent due to the restricted value gates. It should be noted that $s.a.\overline{v}$ is a subset of $\alpha$. Hence, the theorem holds.

**4. Proposed Redundant Identification Method Using Functional Equivalence of Redundant Faults**

In this study, a test generation flow was proposed for transition faults by taking advantage of formal verification tools that could quickly identify the equivalence between almost similar circuit structures including many internal equivalent points. Since the proposed algorithm combined the following known techniques, rapid test generation was available even for hard-to detect transition faults (since the large number of backtracks occurs in commercial ATPG tools and it makes these faults hard-to detect).

1. Modeling a transition fault as a stuck-at fault with temporal expansion
2. Detecting a stuck-at fault by using equivalence checking tools

The input and output of the redundant identification system included the following.

**Input:** a sequential circuit with the full scan design

**Output:** a complete test set that detected all detectable transition faults by using a broadside expansion model

Figure 4 shows the proposed redundant transition fault identification flow. First, in the Step.1, automatic test pattern generation (ATPG) was applied to a given sequential circuit with the full scan design. A commercial ATPG tool was used to identify easy-to detect faults in the circuit to decrease the number of equivalence checking (Step.4). Therefore, only hard-to detect faults were identified as detectable or redundant by applying equivalence checking.

In test generation, ATPG used a low abort limit for each fault (for example, test generation was aborted if it was not completed in 10 s). As a result, a test pattern set was obtained, and fault sets were classified into three types, namely detectable (DT), undetectable (UD), and not detected (ND). Classes DT, UD, and ND implied sets of detected\(^1\) and re-

\(^{1}\)It should be noted that DT was obtained by applying fault simulation with the test pattern set.
dundant faults that were identified by test generation, and a set of faults that were not identified either as detectable or as redundant faults since the abort limit of ATPG was exceeded. Hence, ND was denoted as \( \{ \alpha | \alpha = f_1, f_2, \ldots, f_n \} \), where \( \alpha \) was an element of ND, and there were \( n \) faults in ND.

With respect to Step 2, the temporally expanded circuit with the broadside expansion model \( (C^R) \) was obtained by expanding the sequential circuit with the full scan design. In Step 3, the broadside ATPG model \( (C^R(v)) \) and transition fault injected model \( (C^l(v)) \) were constructed in which \( v \) denotes the restricted value of \( \alpha \). With respect to each fault \( \alpha \) in ND, \( C^R(v) \) and \( C^l(v) \) were denoted as the equivalence check pair and there were \( n \) equivalence check pairs where \( n \) was the size of ND.

In Step 4, equivalence checking was applied for each equivalence check pair for \( n \) times. If \( C^R(v) \) and \( C^l(v) \) were functionally equivalent (Step 5), \( \alpha \) was identified as redundant, and \( \alpha \) was added to the set of redundant transition faults (UD). With respect to the other case, if \( C^R(v) \) and \( C^l(v) \) were not functionally equivalent (Step 6), there existed counter-example patterns that identified this as a potential mismatch. The patterns were also used as a test pattern to detect \( \alpha \).

Therefore, the union of the counter-example pattern set and the ATPG test pattern set corresponded to a complete test set that detected all detectable faults (Step 7). Additionally, the set of transition redundant faults was available from the union set of UD (blocks in Fig. 4 that are marked with diagonal lines) identified by the ATPG tool and equivalence checking tool.

5. Experimental Results

In this section, the proposed redundant fault identification method is evaluated using a commercial ATPG tool by measuring fault efficiency and processing time.

5.1 Experimental Setup

In the experiments, EDA tools were operated on a CAD server (CPU: Intel Core i7-4790K 4GHz, Memory: 32GiB, OS: CentOS 6.8) with ITC’99 benchmark circuit [12] as shown in Table 1. Columns marked as “circuit”, “#PIs”, “#POs”, “#FFs”, “area”, “timing”, and “#faults” indicate the circuit name of each ITC’99 benchmark, the number of primary inputs, the number of primary outputs, the number of flip-flops, the circuit area (A NAND gate is ‘1’), the worst setup path in the circuit, and the number of uncollapsed faults, respectively. The following EDA tools were used for the experiments. Synopsys Design Compiler J-2014.09-SP1, Tetra Max J-2014.09-SP1, and Formality A-2008.03-SP3 were used as design optimization and full scan design, test generation, and equivalence checking, respectively. Design Optimization was applied with a flatten option and test generation was applied with nopi_changes, add_po_masks and without reset in the test mode (scan shift and launch-capture).

5.2 Formal Verification Approach vs ATPG

In this subsection, the proposed redundant fault identification method was evaluated against a commercial ATPG tool with respect to fault efficiency and processing time. Fault efficiency (FE) is widely used as an evaluation criteria for test generation, and it shows the ratio of the number of detected and redundant faults identified by a target test generation algorithm for all faults, defined as follows.

\[
FE(\%) = \frac{|DT| + |UD|}{#faults} \times 100
\]

where \( |DT| \) and \( |UD| \) denote the number of detectable faults (size for the set of DT) and the number of redundant faults (size for the set of UD), respectively. Additionally, \( #faults \) refers to the number of all faults in the circuit. Furthermore, ATPG time limitation (abort limit) that was used in the proposed redundant identification method was determined as follows:

1. Max abort limit under 10 s spent in ATPG
2. Default abort limit value (abort limit = 10 s per 1 fault) if ATPG did not finish in 10 s

Table 2 shows the experimental results for fault efficiency and CPU processing time. Columns “AL”, “#ND”, “ATPG”, and “[%]” under “Formal approach” denote abort limit used to ATPG in proposal, the number of uncollapsed faults that were not identified as either detectable or as redundant given that the abort limit was exceeded, the CPU processing time (in seconds) of ATPG, and its ratio of “Total time”, respectively. Columns “TE” and “[%]” denote CPU processing time of temporal expansion and its ratio for “Total time”. In a similar manner, columns “EC” and “[%]” indicate CPU processing time of equivalence checking and its ratio, respectively. Columns “FE[%]” and “Total time” show fault efficiency of the proposed method and

| Circuit | #PI | #PO | #FFs | Area | Timing | #faults |
|--------|-----|-----|------|------|--------|---------|
| b01    | 2   | 2   | 5    | 50   | 5.12   | 204     |
| b02    | 1   | 1   | 4    | 61   | 4.82   | 136     |
| b03    | 4   | 4   | 30   | 430  | 10.71  | 784     |
| b04    | 11  | 8   | 66   | 660  | 27.71  | 3,102   |
| b05    | 1   | 36  | 34   | 1,509| 31.16  | 5,110   |
| b06    | 2   | 6   | 9    | 142  | 6.82   | 304     |
| b07    | 1   | 8   | 44   | 856  | 18.78  | 1,972   |
| b08    | 9   | 4   | 21   | 350  | 10.98  | 806     |
| b09    | 1   | 1   | 28   | 419  | 11.94  | 844     |
| b10    | 11  | 6   | 17   | 325  | 9.59   | 828     |
| b11    | 7   | 6   | 31   | 1,001| 22.37  | 2,994   |
| b12    | 5   | 6   | 121  | 2,089| 16.33  | 4,902   |
| b13    | 10  | 10  | 52   | 778  | 8.91   | 1,509   |
| b14    | 32  | 54  | 215  | 12,998| 98.66  | 39,838  |
| b15    | 36  | 70  | 418  | 11,441| 98.75  | 33,110  |
| b17    | 37  | 97  | 1,320| 35,453| 98.75  | 101,570 |
| b20    | 32  | 32  | 430  | 26,853| 98.75  | 81,746  |
| b21    | 32  | 32  | 430  | 27,024| 98.75  | 82,026  |
| b22    | 32  | 32  | 613  | 39,990| 98.75  | 121,220 |

Table 1: Circuit specification of ITC’99 benchmark circuits.
sumption of “ATPG”, “TE”, and “EC” (the unit was formatted), respectively. Columns “AL”, “FE[%]”, and “time” under “ATPG” correspond to the abort limit to achieve the best fault efficiency with ATPG, its fault efficiency, and CPU processing time of ATPG with the abort limit (the unit was formatted), respectively. The experiments indicated that the proposed redundant identification method achieved complete fault efficiency (100%) within a reasonable time for the all circuits used in the experiments. However, there were several circuits in which ATPG did not achieve 100% fault efficiency within a week.

b01-03 and b06-13 indicated the same results from ATPG because ATPG was completed within 10 s.

Table 2: Experimental results for fault efficiency and CPU processing time.

| Target | Formal approach | Summary | ATPG |
|--------|-----------------|---------|------|
|        | AL   | #ND | ATPG [%] | CPU processing time[s] | AL | FE[%] | Total time |
| b01    | 100  | 0   | 0.16  | 100 | 0.16[s] | 100 | 0.16[s] |
| b02    | 10   | 0   | 0.16  | 100 | 0.16[s] | 100 | 0.16[s] |
| b03    | 1K   | 0   | 0.17  | 100 | 0.17[s] | 1K  | 0.17[s] |
| b04    | 10K  | 197 | 4.35  | 100 | 0.17[s] | 1K  | 0.17[s] |
| b06    | 100  | 0   | 0.16  | 100 | 0.16[s] | 100 | 0.16[s] |
| b07    | 10K  | 0   | 0.94  | 100 | 0.20[s] | 10K | 0.20[s] |
| b08    | 10K  | 0   | 0.27  | 100 | 0.140[s] | 10K | 0.140[s] |
| b09    | 10K  | 0   | 1.40  | 100 | 0.94[s] | 10K | 0.94[s] |
| b10    | 1K   | 0   | 0.20  | 100 | 0.27[s] | 1K  | 0.27[s] |
| b11    | 10K  | 0   | 1.56  | 100 | 0.140[s] | 10K | 0.140[s] |
| b12    | 10K  | 0   | 7.44  | 100 | 7.44[s] | 10K | 7.44[s] |

Table 3: Fault classification results.

| #faults | Formal approach | EC results | ATPG (Best FE) |
|---------|-----------------|------------|----------------|
|         | #DT | #UD | #ND | #DT | #UD | #ND | #ND(DT) | #ND(UD) |
| b01     | 204 | 0   | 0   | 138 | 66  | 0   | -       | -       |
| b02     | 136 | 0   | 0   | 109 | 27  | 0   | -       | -       |
| b03     | 784 | 0   | 0   | 707 | 77  | 0   | -       | -       |
| b04     | 3,102 | 0   | 49  | 0   | 9   | 0   | 1,921   | 1,181   |
| b05     | 5,110 | 197 | 9   | 0   | 1,491 | 3,619   | -       | -       |
| b06     | 304 | 0   | 0   | 183 | 121 | 0   | -       | -       |
| b07     | 1,972 | 0   | 0   | 1,567 | 405 | 0   | -       | -       |
| b08     | 806 | 0   | 0   | 623 | 183 | 0   | -       | -       |
| b09     | 844 | 0   | 0   | 727 | 117 | 0   | -       | -       |
| b10     | 828 | 0   | 0   | 541 | 287 | 0   | -       | -       |
| b11     | 2,994 | 0   | 0   | 0   | 0   | 0   | -       | -       |
| b12     | 4,902 | 0   | 0   | 0   | 0   | 0   | -       | -       |
| b13     | 1,556 | 0   | 0   | 0   | 0   | 0   | -       | -       |
| b14     | 39,838 | 5,797 | 4,171 | 1,626 | 0   | 36,826 | 1,166 | 1,846 | 708 | 1,138 |
| b15     | 33,110 | 7,939 | 2,764 | 5,175 | 0   | 26,034 | 6,216 | 860  | 89  | 771  |
| b17     | 101,570 | 79,175 | 18,688 | 6,087 | 12,601 | 0   | 84,773 | 14,057 | 2,740 | 489 | 2,251 |
| b20     | 81,746 | 74,429 | 5,849 | 2,934 | 2,915 | 0   | 76,040 | 2,045 | 3,361 | 1,323 | 2,338 |
| b21     | 82,026 | 72,547 | 8,009 | 5,119 | 2,890 | 0   | 74,900 | 2,027 | 5,099 | 2,766 | 2,333 |
| b22     | 121,220 | 118,304 | 5,102 | 767 | 4,335 | 0   | 114,304 | 3,196 | 3,720 | 267 | 3,453 |

pos and ATPG since the number of ND was not small (197 faults). With respect to b14, b15, b20, b21, and b22, complete fault efficiency was obtained within 15 h (results were be available at 9AM on the following day if ATPG commenced at 6PM on the previous day). With respect to b17, the proposal required two days to achieve the complete fault efficiency since #ND exceeded that of the other circuits. Conversely, ATPG did not achieve 100% fault efficiency for these circuits (a month was required to obtain higher fault efficiency by advancing the abort limit).

Table 3 shows the number of faults classified by each approach. In the proposed formal approach, equivalence checking was applied for the ND faults classified with a light abort limit ATPG. Columns “#DT”, “#UD”, and “#ND” under “ATPG (light) result” indicated the number of uncollapsed detectable faults identified with light abort limit
ATPG, redundant faults, and not detected faults, respectively. Similarly, columns marked as “EC result” show the results of faults identified with the proposed redundant identification method. The columns “#DT”, “#UD”, and “#ND” under “ATPG (Best FE)” correspond to the results of ATPG in which the best fault efficiency was achieved. The numbers of detectable and redundant faults in ND were calculated by using the results of the proposed formal approach. Columns “#ND(DT)” and “#ND(UD)” indicated the number of detectable and redundant faults in ND, respectively. The formal approach and ATPG classified all faults using the same classification for the circuits in which ATPG achieved 100% fault efficiency. Both time-consuming redundant faults as well as several detectable faults existed in the faults that could not be identified with no limit ATPG.

5.3 Experimental Results for Larger Circuits

Figure 5 shows the stacked chart for the ratio of CPU processing time for each step in the proposed approach for larger circuits that were not finished by only using light abort limit ATPG (b04, b05, b14, b15, b17, b20, b21, and b22). The stack order from the bottom corresponded to the time required for light abort limit ATPG, temporal expansion, and equivalence checking. The ratio of light abort limit ATPG was very low for larger circuits. The ratio of temporal expansion also decreased with increases in the area scale. Therefore, it was observed that CPU processing time was influenced by equivalence checking.

5.4 Statistical Analysis for Each Fault

Table 4 shows statistical analysis of the results of the proposed redundant identification method without using light abort limit ATPG for each collapsed fault in 13 small circuits (b01-b13). Columns “TE” and “EC” show CPU processing time (in seconds) required for temporal expansion and for equivalence checking, respectively. Column “Total[sec]” shows the total CPU processing time (in seconds) obtained by the summation of TE and EC. Columns “Avg”, “σ”, “Best”, and “Worst” under “each fault” indicate the average of CPU processing time (in seconds) for each fault, population variance, the fastest and slowest identification, respectively. The results indicated that the time variation for the faults was minimal since the result of population variance was small.

Table 5 shows the statistical analysis of CPU processing time for redundant identification with proposal and ATPG for each collapsed fault that is not identified with light abort limit ATPG. In the experiments, b04 and b05 were used as evaluation targets since light abort limit ATPG could achieve 100% fault efficiency. With respect to b04 and b05, the remaining collapsed faults (Column “#ND”) corresponded to 29 and 171 with abort limit 10,000 and 1,000, respectively. With respect to the remaining faults, redundant identification was applied by using the proposal and ATPG with increasing abort limit (Column “AL”) to 1 billion (b04) and 1 million (b05). Columns “Time[sec]”, “Avg”, “σ”, “Best”, and “Worst” under “Formal approach” and “ATPG” correspond to the total CPU processing time (seconds), average, population variance, the fastest identification, and slowest identification, respectively. The result revealed that the CPU processing time of ATPG resulted in large variations. Conversely, only a small variation existed in the case of the proposed formal approach. Therefore, it could be concluded that the proposed approach possessed the advantage of “on-time” execution.

Table 6 shows the statistical analysis of CPU processing time for the proposed redundant identification method for each collapsed fault that could not be identified with light abort limit ATPG. In the experiments, only the result of pro-

![Fig. 5 Stacked chart for the ratio of CPU time.](image-url)

| #ND | Formal approach | ATPG |
|-----|----------------|------|
|     | Time[sec] | Avg  | σ    | Best | Worst | AL | Time[sec] | Avg  | σ    | Best | Worst |
| b04 | 20.24  | 19.25 | 0.66 | 0.25 | 0.005 | 0.23 | 0.26 |
| b05 | 191.16 | 114.62 | 5.10 | 0.50 | 0.035 | 0.44 | 0.60 |
| b04 | 1.31182 | 776.54 | 34.81 | 0.87 | 0.070 | 0.64 | 1.13 |
| b05 | 2.2657 | 1.7382 | 57.33 | 0.87 | 0.082 | 0.60 | 1.24 |
| b04 | 45.91 | 43.27 | 1.49 | 0.36 | 0.029 | 0.29 | 0.45 |
| b05 | 581.24 | 429.27 | 16.84 | 0.69 | 0.054 | 0.52 | 0.90 |
| b04 | 178.56 | 135.03 | 5.23 | 0.49 | 0.030 | 0.39 | 0.66 |
| b05 | 198.59 | 141.08 | 5.66 | 0.53 | 0.034 | 0.43 | 0.69 |
| b04 | 183.64 | 141.27 | 5.42 | 0.49 | 0.027 | 0.41 | 0.62 |
| b05 | 1,009.84 | 632.28 | 27.37 | 0.72 | 0.038 | 0.63 | 0.85 |
| b04 | 3,159.95 | 1,632.62 | 79.88 | 1.22 | 0.076 | 1.00 | 1.49 |
| b05 | 480.56 | 259.00 | 12.33 | 0.59 | 0.022 | 0.51 | 0.68 |

Table 4 Redundancy identification for each collapsed fault.
The proposed redundant identification method was reflected since ATPG was not completed within a month for the remaining larger six circuits (b14, b15, b17, b20, b21, and b22). The result revealed that the proposed approach also possessed the advantage of “on-time” execution with respect to the larger circuits.

6. Conclusion

Nano-scaling demands an extremely reliable manufacturing test for delay faults. However, it is not easy to achieve 100% fault efficiency for combinational circuits based on the full scan design for large practical circuits. In order to identify whether a fault is redundant, it is necessary for ATPG to repeat large numbers of backtracks with respect to a fault, and it leads to indefinite CPU processing time for test generation. Thus, in the present study, a test generation flow was proposed for transition faults by taking advantage of formal verification tools that could quickly identify the equivalence between almost similar circuit structures including many internal equivalent points. Functional equivalence for redundant stuck-at faults is a widely-known topic. However, in the present study, the functional equivalence for redundant transition faults was defined based on a broadside expansion model. The experimental results indicated that the benefit of the proposed redundant identification method using a formal approach included “on-time” execution.

The application of the proposed method was related to an existing issue of identification for sequentially redundant transition faults. Thus, the benefits include increasing manufacturing test reliability as well as alleviating over-testing and improving yield.

Acknowledgments

The authors thank Dr. Thomas Edison Yu for his insightful comments and suggestions. This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] “International technology roadmap for semiconductors 2011 edition: Design,” 2011.
[2] M. Fujita, “Guide to verification techniques for system LSI—equivalence checking for logical circuits—,” Design Wave Magazine, Feb. 2004. (in Japanese).
[3] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S. Kajihara, “Invisible delay quality—SDQM model lights up what could not be seen,” Proc. International Test Conference, 47.1, 2005.
[4] K.-T. Cheng and H.-C. Chen, “Classification and identification of nonrobust untestable path delay faults,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.15, no.8, pp.845–853, Aug. 1996.
[5] S. Kajihara, K. Kinoshita, I. Pomeranz, and S.M. Reddy, “A method for identifying robust dependent and functionally unsensitizable paths,” Proc. International Conference on VLSI Design, pp.82–87, Jan. 1997.
[6] Y. Shao, S.M. Reddy, S. Kajihara, and I. Pomeranz, “An efficient method to identify untestable path delay faults,” Proc. Asian Test Symposium, pp.233–238, 2001.
[7] J. Moondanos and J. Abraham, “Sequential redundancy identification using verification techniques,” Proc. International Test Conference, pp.197–205, 1992.
[8] M. Abramovici and M.A. Iyer, “Method for identifying untestable faults in logic circuits,” US Patent 5,566,187, Oct. 1996.
[9] D. Moondanos, J.A. Abraham, and Y.V. Hoskote, “Abstraction techniques for validation coverage analysis and test generation,” IEEE Trans. Comput., vol.47, no.1, pp.2–14, Jan. 1998.
[10] I. Pomeranz, “Fast identification of undetectable transition faults under functional broadside tests,” IEEE Trans. Comput., vol.61, no.6, pp.905–910, 2012.
[11] X. Liu and M.S. Hsiao, “A novel transition fault ATPG that reduces yield loss,” IEEE Des. Test. Comput., vol.22, no.6, pp.576–584, 2005.
[12] F. Corno, M.S. Reorda, and G. Squillero, “Rt-level itc’99 benchmarks and first atpg results,” IEEE Des. Test. Comput., vol.17, no.3, pp.44–53, 2000.

Hiroshi Iwata received his B.E. degree in information engineering from National Institute of Technology, Nara College, Japan, in 2007 and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 2008 and 2011, respectively. His research interests include VLSI CAD, design for testability, and asynchronous circuit testing. He is a member of IEEE.

Nanami Katayama received her B.E. degree in information engineering from National Institute of Technology, Nara College, Japan, in 2016. Her research interests include embedded system, VLSI testing and formal verification.

Ken’ichi Yamaguchi received his B.E. degree in information engineering from National Institute of Technology, Nara College, Japan, in 1999 and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology, Nara, Japan, in 2001 and 2003, respectively. His research interests include VLSI CAD, design for testability, and high-level synthesis.

Table 6 Identification results for hard-to-detect faults.

| # | ND | Avg | σ | Best | Worst |
|---|----|-----|---|------|-------|
| b14 | 4,631 | 4.83 | 0.454 | 3.32 | 6.66 |
| b15 | 6,451 | 4.60 | 0.543 | 3.30 | 7.49 |
| b17 | 15,154 | 11.77 | 0.633 | 9.71 | 14.89 |
| b20 | 4,872 | 8.17 | 0.688 | 6.03 | 12.29 |
| b21 | 6,396 | 8.24 | 0.705 | 6.15 | 38.41 |
| b22 | 4,279 | 10.52 | 0.869 | 8.69 | 20.10 |