Logic/Clock-Path-Aware At-Speed Scan Test Generation for Avoiding False Capture Failures and Reducing Clock Stretch

K. Asada [1], X. Wen [1], S. Holst [1], K. Miyase [1], S. Kajihara [1], M. A. Kochte [2], E. Schneider [2], H.-J. Wunderlich [2], and J. Qian [3]

[1] Kyushu Institute of Technology, Iizuka, 820-8502, Japan
[2] University of Stuttgart, Pfaffenwaldring 47, 70569 Stuttgart, Germany
[3] Advanced Micro Devices, Inc., Sunnyvale, CA 94088, USA

Abstract: *IR-drop induced by launch switching activity (LSA) in capture mode during at-speed scan testing increases delay along not only logic paths (LPs) but also clock paths (CPs). Excessive extra delay along LPs compromises test yields due to false capture failures, while excessive extra delay along CPs compromises test quality due to test clock stretch. This paper is the first to mitigate the impact of LSA on both LPs and CPs with a novel LCPA (Logic/Clock-Path-Aware) at-speed scan test generation scheme, featuring (1) a new metric for assessing the risk of false capture failures based on the amount of LSA around both LPs and CPs, (2) a procedure for avoiding false capture failures by reducing LSA around LPs or masking uncertain test responses, and (3) a procedure for reducing test clock stretch by reducing LSA around CPs. Experimental results demonstrate the effectiveness of the LCPA scheme in improving test yields and test quality.*

Keywords: *launch switching activity, IR-drop, logic path, clock path, false capture failure, test clock stretch, X-filling*

1. Introduction

At-speed scan testing is indispensable for cost-efficiently testing logic LSI circuits for timing-related defects [1] and also helpful for speed binning [2]. Test vectors for at-speed scan testing are usually obtained from ***automatic test pattern generation*** (***ATPG***) for transition and/or path delay fault models. To cope with the growing dominance of small-delay defects in deep-submicron LSIs [3], timing-aware ATPG, which tries to sensitize longer paths for fault effect propagation, has also come into use [4].

The principle of at-speed scan testing is to launch a transition at the start-point of a ***logic path*** (i.e., the *Q*-output of a flip-flop) and capture its response at the end-point of the logic path (i.e., the *D*-input of a flip-flop) at the functional clock speed [1,2]. This is illustrated in Fig. 1, in which the ***launch-on-capture*** (***LOC***) clocking scheme is used. Here, a transition is launched from the start-point (FF_S) of the logic path *lp* by the first capture clock pulse C_1 and the response to the transition is captured at the end-point (FF_e) of *lp* by the response capture clock pulse C_2. Note that "*at-speed*" requires that the test clock period *T* be set to the functional clock period *Tf*. This is to guarantee that excessive delay increase along *lp* can be detected as a timing failure by the response capture clock pulse C_2.

It is well-known that the ***launch switching activity*** (***LSA***) triggered by transition launch at C_1 in at-speed scan testing, is much higher than that in function mode [5-9]. Excessive LSA can induce severe IR-drop, which may cause two major problems, namely ***false capture failure*** and ***test clock stretch***, as illustrated in Fig. 1.

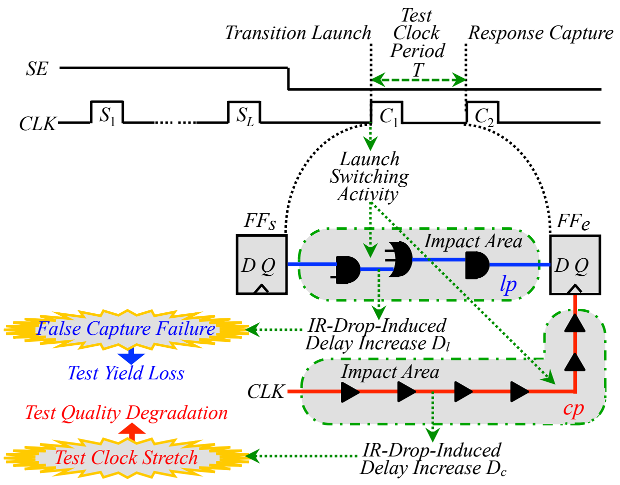


Fig. 1. Impact of LSA in LOC-based at-speed scan testing.

- **Problem-1** (***False Capture Failure***): *Launch switching activity* triggered at transition launch (C_1) occurs in the neighborhood (***impact area***) of the logic path *lp*, causing IR-drop at its on-path gates and thus resulting in an extra delay D_l along *lp*. If D_l is larger than the slack of *lp*, a ***false capture failure*** will occur at the endpoint (FF_e) of *lp* at response capture (C_2) for a defect-free circuit, causing ***test yield loss***. This is a severe over-testing problem, especially for high-speed /low-power LSI circuits [5-9].
- **Problem-2** (***Test Clock Stretch***): *Launch switching activity* triggered at transition launch (C_1) also occurs in the impact area of the clock path *cp* corresponding to the logic path *lp*. This causes IR-drop at the on-path clock buffers of *cp* and results in an extra delay D_c along *cp*. The test clock period *T* now becomes $Tf + D_c$, where *Tf* is the functional clock period. This phenomenon is known as ***test clock stretch*** [10, 11], which makes testing slower (e.g., as much as 15%) than "*at-speed*". This is a major cause for test quality degradation due to test escape, especially for deep-submicron LSI circuits with small-delay defects.

Problem-1 (*False Capture Failure*) has been tackled with various low capture power solutions through ***circuit/ clock modification*** and/or ***test data manipulation*** [5-9]. Recently, ***pinpoint*** solutions for achieving ***capture safety*** (i.e., assurance that no LSA-induced false capture failure will occur in at-speed scan testing) have been proposed [12, 13]. A capture-safe solution first identifies ***risky paths*** (i.e., any long sensitized logic path with excessive LSA in its neighbourhood) and then reduces the local LSA. If the local LSA cannot be reduced under a safe level, the test response bit corresponding to the end-point of the risky path is then masked so as to prevent the possibly false test response from causing a wrong test decision on the tester. The advantage of such a capture-safe solution is that it can completely avoid false capture failures with negligible test data inflation [12, 13]. However, previous capture-safe solutions lack accuracy since they only consider the impact of LSA on logic paths but ignore clock paths in risky path checking. This is a severe issue since clock stretch due to the impact of LSA on clock paths may increase a test clock period by as much as 15% [10, 11]. *Therefore, the first goal of this work is to improve the accuracy of risky path checking for more efficiently avoiding false capture failures by considering both logic and clock paths*.

Problem-2 (*Test Clock Stretch*) is a difficult-to-handle issue due to its ***local*** and ***dynamic*** nature. That is, the amount of test clock stretch for a *flip-flop* (*FF*) depends on its location, the neighborhood (or impact area) of its clock path, the *power distribution network* (*PDN*) design, the test vector applied, etc. A calibration-based solution [10] has been proposed, in which measurement blocks are inserted into a few locations in a circuit for clock stretch assessment and test guard bands are adjusted accordingly. This solution is ***global*** and ***static*** (i.e., test clock stretch cannot be mitigated on a per-FF/per-vector basis), making it less accurate. In addition, this solution is costly since it needs to insert on-chip measurement blocks. *Therefore, the second goal of this work is to provide a local, dynamic, and low-cost solution for reducing test clock stretch*.

In order to achieve the above-mentioned two goals, this paper proposes a novel ***LCPA*** (***Logic/Clock-Path-Aware***) at-speed scan test generation scheme, featuring (1) a new metric for risky path checking based on LSA around both ***logic paths*** and ***clock paths***, (2) a procedure for avoiding false capture failures by first ***rescuing*** (i.e., reducing local LSA around risky paths) and then ***masking*** (i.e., instructing a tester to ignore the test response from any remaining risky path), and (3) a procedure for mitigating test clock stretch on a per-FF/per-vector basis by reducing LSA around clock paths for each test vector. The major contributions of this paper are as follows:

(1) More accurate metric for risky path checking leads to fewer risky paths, resulting in less test data inflation associated with avoiding false capture failures.

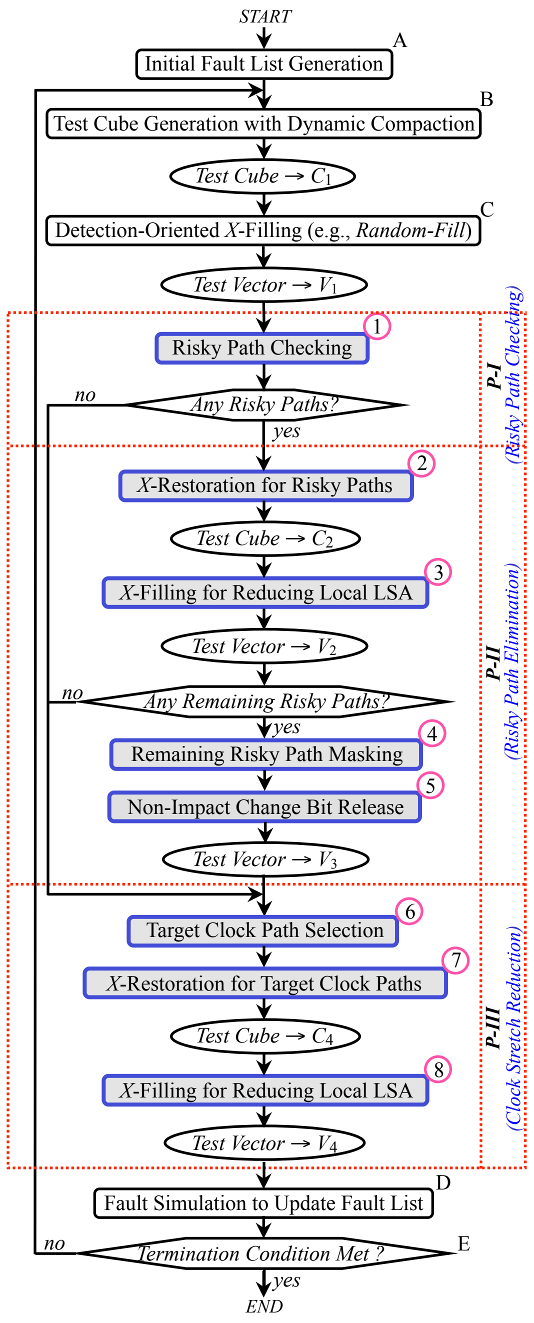
(2) The first ***pinpoint*** (per-FF/per-vector) technique for reducing clock stretch by test data manipulation instead of circuit modification, resulting in a local, dynamic, and low-cost solution for mitigating test clock stretch.

The rest of the paper is organized as follows: Sect. 2 outlines the general LCPA test generation scheme; Sect. 3 describes the details of the LCPA scheme; Sect. 4 shows experimental results, and Sect. 5 concludes the paper.

2. General Flow

Fig. 2 shows the general flow of the proposed LCPA test generation scheme, which consists of conventional test generation steps (A~E) and new steps (①~⑧).

Conventional test generation starts from *initial fault list generation* (A). Each test generation run begins with an all-*X* input cube, and logic values are gradually assigned to the *X*-bits to detect a *primary fault* and optionally *secondary faults* through *dynamic compaction* (B). For the



START
A
Initial Fault List Generation
B
Test Cube Generation with Dynamic Compaction
Test Cube → C1
C
Detection-Oriented X-Filling (e.g., Random-Fill)
Test Vector → V1
① Risky Path Checking
Any Risky Paths?
no
yes
P-I (Risky Path Checking)
② X-Restoration for Risky Paths
Test Cube → C2
③ X-Filling for Reducing Local LSA
Test Vector → V2
Any Remaining Risky Paths?
no
yes
④ Remaining Risky Path Masking
⑤ Non-Impact Change Bit Release
Test Vector → V3
P-II (Risky Path Elimination)
⑥ Target Clock Path Selection
⑦ X-Restoration for Target Clock Paths
Test Cube → C4
⑧ X-Filling for Reducing Local LSA
Test Vector → V4
P-III (Clock Stretch Reduction)
D
Fault Simulation to Update Fault List
E
Termination Condition Met ?
no
yes
END

Fig. 2. General flow of the LCPA test generation scheme.

resulting partially-specified test cube C_1, detection-oriented *X*-filling (usually *random-fill*) is conducted (C), resulting in a fully-specified test vector V_1. In conventional test generation, V_1 is the final test vector for the current run. Fault simulation is then conducted to update the fault list (D), and the termination condition is checked to determine whether to continue test generation (E).

The proposed LCPA test generation scheme shown in Fig. 2 enhances conventional test generation (A~E) with new steps (①~⑧) that form three phases, namely ***P-I*** (*Risky Path Checking*), ***P-II*** (*Risky Path Elimination*), and ***P-III*** (*Clock Stretch Reduction*). Their details will be described in Subsections 3.1, 3.2, and 3.3, respectively.

3. Details of the LCPA Scheme

3.1 Risky Path Checking

Risky Path Checking (P-I: ① in Fig. 2) identifies every logic path sensitized by a test vector V that may cause a false capture failure due to excessive *launch switching activity* (*LSA*) in its neighborhood. Obviously, only "***long***" logic paths are susceptible to the impact of LSA in term of LSA-induced extra path delay [9, 12, 13].

Definition 1: A ***long sensitized path*** (***LSP***) P of a test vector V is a logic path that is sensitized by V and whose slack is smaller than the maximum LSA-induced extra delay along P.

Clearly, a long path P with a *positive* slack in function-mode may have a *negative* slack in scan capture mode due to the excessive LSA-induced extra delay along P. This is because LSA-induced IR-drop at the on-path gates of P can be much higher in scan capture mode than in function mode. To avoid time-consuming circuit simulation and IR-drop/delay analysis, some static approximation thresholds, such as a percentage (e.g., 20%) of the test cycle time for checking the nominal slack of P, or, a percentage (e.g., 70%) of the maximum path delay in the circuit for checking the nominal path delay of P, are often used in practice for determining whether P is long or not [9, 12, 13].

Definition 2: Suppose that an LSP of a test vector V has a nominal delay of D_n, an LSA-induced delay of D_l, and the test clock period is T. The LSP is a ***risky path*** of V if $D_l > (T - D_n)$; otherwise, the LSP is a ***safe path*** of V.

Previous risky path checking methods [12, 13] assume ***ideal*** at-speed scan testing, in which $T = T_f$ for any LSP, where T_f is the functional clock period for the circuit-under-test. In this case, an LSP is considered as a risky path if

$$D_l > (T_f - D_n) \qquad \text{(E1)}$$

However, the condition E1 only considers the LSA-inducted IR-drop impact on an LSP (i.e., D_l in Fig. 1). However, in ***real*** at-speed scan testing, the clock path of the end-point FF of an LSP is also affected by LSA-inducted IR-drop, resulting in test clock stretch for the FF [10, 11]. Clearly, this test clock stretch (denoted by D_c in Fig. 1) also needs to be taken into consideration, resulting in $T = T_f + D_c$. That is, in real at-speed scan testing, an LSP should be considered as a risky path if

$$(D_l - D_c) > (T_f - D_n) \qquad \text{(E2)}$$

Clearly, the logic/clock-path-aware condition E2 is more accurate for risky path checking. This is because, as shown in Fig. 3, ignoring test clock stretch (D_c) will result in pessimism that overestimates a safe path as a risky path.

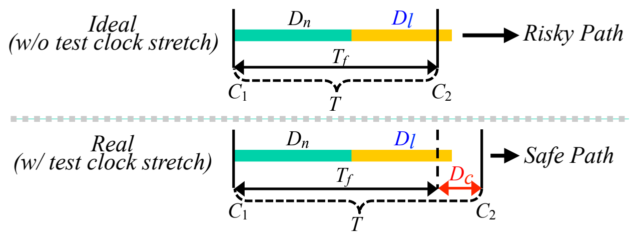


Fig. 3. Pessimism due to ignoring test clock stretch.

Since the direct use of the condition E2 for risky path checking entails costly circuit simulation and IR-drop/delay analysis, this paper proposes a new approximation metric, as a form of ***weighted switching activity*** (***WSA***) [7-9], for scalable and efficient risky path checking based on the condition E2. The basic idea is to calculate *WSA* values in the neighborhoods of an LSP and its clock path as estimates for LSA-induced extra delay values D_l and D_c along the LSP and its clock path, respectively. The neighborhood of a path is defined as its "*impact area*" [12, 13] as follows:

Definition 3: The ***aggressor region*** of a gate G, denoted by $AR(G)$, is composed of logic elements (gates and FFs) whose transitions strongly impact the supply voltage of G. The ***impact area*** of a path P, denoted by $IA(P)$, consists of the aggressor regions of all of its on-path gates (G_1, G_2, ..., G_P). That is, $IA(P) = AR(G_1) \cup AR(G_2) \cup \ldots \cup AR(G_P)$.

Fig. 4 illustrates the impact area of a logic or clock path P, which can be readily identified with data of layout and *power distribution network* (*PDN*) design [9, 12, 13].

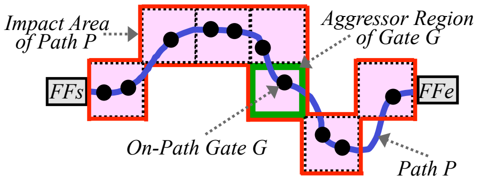


Fig. 4. Impact area of a path.

As illustrated in Fig. 5, assume that V is a test vector, *lsp* is a *long sensitized path* (*LSP*) of V and *cp* is the clock path of *lsp*. Denote the WSA for the impact area of *lsp* under V, the WSA for the impact area of *cp* under V, and the maximum WSA for the impact area of *lsp* by ***WSA***(***V***, ***lsp***), ***WSA***(***V***, ***cp***), and ***WSAmax***(***lsp***), respectively. From the result of logic simulation for V, these three WSA values can be calculated as follows:

$$WSA(V, lsp) = \sum_{ei \in IA(lsp)} (W(ei) * T(V, ei))$$

$$WSA(V, cp) = \sum_{ei \in IA(cp)} (W(ei) * T(V, ei))$$

$$WSAmax(lsp) = \sum_{ei \in IA(lsp)} W(ei)$$

Here, e_i is a logic element in $IA(lsp)$ or $IA(cp)$, $W(e_i)$ is the fanout count of e_i, $T(V, e_i) = 1$ (0) if launch switching activity induced by V causes (does not cause) a transition at e_i. Note that $WSAmax(lsp)$ is calculated by assuming that all logic elements in $IA(lsp)$ have transitions.

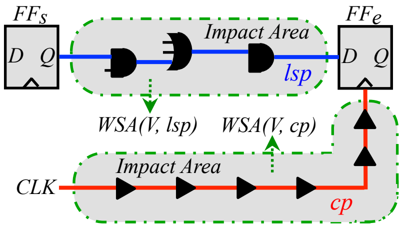


Fig. 5. Logic/logic-path-aware risky path checking.

The new metric for determining whether the LSP *lsp* is a risky path of the test vector V is as follows:

$$WSA(V, lsp) - WSA(V, cp) > \alpha * WSAmax(lsp) \quad \text{(E3)}$$

Comparing E2 and E3 reveals the reasoning behind the new metric E3 for risky path checking as follows: First, $WSA(V, lsp)$ and $WSA(V, cp)$ are estimates for D_l and D_c, respectively; $WSAmax(lsp)$ is an estimate for the maximum value of D_l. Since *lsp* is a long sensitized path, the maximum value of $D_l > (T_f - D_n)$ according to Definition 1. Therefore, $\alpha * WSAmax(lsp)$, where $\alpha < 1$, can be used as an estimate for $(T_f - D_n)$. In practice, α can be determined through simulation experiments with functional vectors or set as a parameter as in commercial EDA tools.

Given a circuit *CUT* and a test vector *V*, logic/clock-path-aware risky path checking is conducted as follows:

```
(1) Conduct logic simulation for V on CUT.
(2) Identify all LSPs of V.
(3) For each LSP lsp and its corresponding
    clock path cp, identify their impact
    areas, calculate WSA(V, lsp), WSA(V, cp),
    WSAmax(lsp), and  use the metric E3 to
    determine whether lsp is risky.
```

As the first *LCPA* (*Logic/Clock-Path-Aware*) metric, the condition E3 considers LSA-induced extra delay along not only logic paths but also clock paths. This results in higher accuracy for risky path checking since test clock stretch is too significant to be ignored [10, 11].

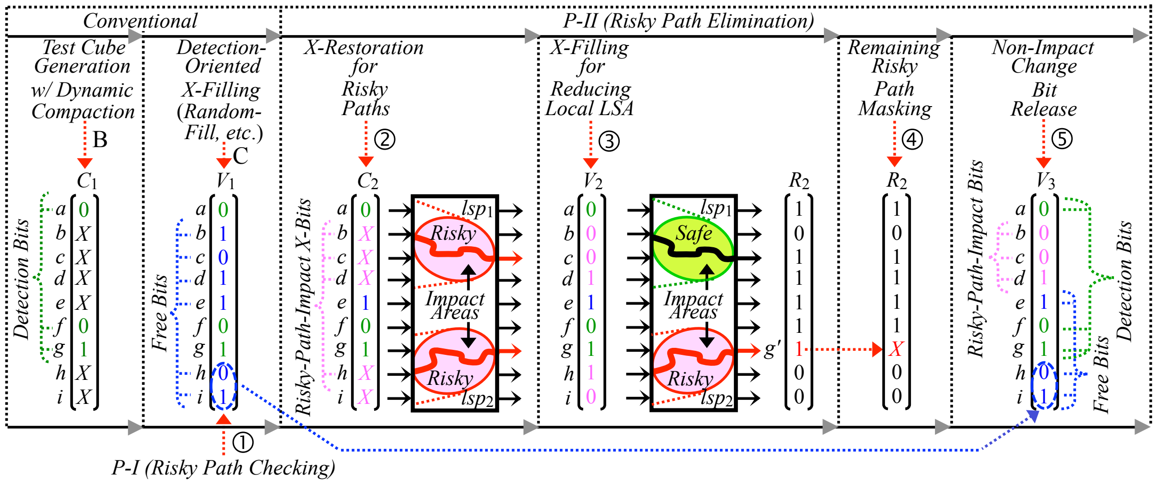
3.2 Risky Path Elimination

Risky Path Elimination (*P-II*: ②~⑤ in Fig. 2) is to modify a test vector with risky paths so as to prevent them from causing any false capture failures while maintaining its fault detection capability as much as possible. It is enhanced over our previous work [12, 13] by adding a new step (⑤) for further reducing test data inflation.

In the example of Fig. 6, a partially-specified test cube C_1 is first generated solely for fault detection without any test power consideration, whose specified bits are for detecting a primary fault and optionally secondary faults in dynamic compaction (B). Such specified bits are referred to as ***detection bits*** hereafter. Detection-oriented *X*-filling (such as *random-fill*) is then conducted to turn C_1 into a fully-specified test vector V_1, whose newly specified logic values help in fortuitous fault detection (C). Such newly specified logic values are referred to ***free bits*** hereafter. After that, *P-I* (*Risky Path Checking*) is conducted on V_1 (①). Suppose that lsp_1 and lsp_2 are identified as risky paths. *P-II* (*Risky Path Elimination*) is then conducted through three stages (***rescue***, ***mask***, ***release***), as follows:

- **Stage-1 (*Rescue*)**: ***X-restoration for risky paths*** (②) is conducted to identify the free bits in V_1 that can reach the impact areas of lsp_1 and lsp_2, and then turn those bits back into *X*-bits (referred to as *risky-path-impact X-bits*). The result is a new partially-specified test cube C_2. After that, ***X-filling for reducing local LSA*** (③) is conducted on risky-path-impact *X*-bits in C_2 for reducing *launch switching activity* (*LSA*) in the impact areas of lsp_1 and lsp_2. This results in a new fully-specified test vector V_2 with reduced $WSA(V_2, lsp_1)$ and $WSA(V_2, lsp_2)$. In this example, $WSA(V_2, lsp_1)$ is sufficiently reduced, turning lsp_1 into a safe path according to the metric E3; however, lsp_2 remains a risky path since $WSA(V_2, lsp_2)$ is not sufficiently reduced.
- **Stage-2 (*Mask*)**: lsp_2 being risky under V_2 means that the bit *g'* in the test response R_2 corresponding to the endpoint of lsp_2 may become 0 instead 1, possibly resulting in a false capture failure. To avoid this possibility, ***remaining risky path masking*** (④) is conducted by placing a masking symbol (e.g., *X*) at *g'* in R_2. Note that ***mask*** guarantees capture power safety without adding any new circuitry.
- **Stage-3 (*Release*)**: In Fig. 6, the initial fault-detection test vector V_1 has *free bits* 0 and 1 at *h* and *i*, respectively, which helps for fortuitous fault detection. Later these two free bits are restored as *risky-path-impact X-bits* and filled with 1 and 0, respectively, in V_2 for "rescuing" lsp_2. Since this rescue effort fails to turn lsp_2 into a safe path, ***non-impact change bit release*** (⑤) is then conducted to change the *X*-filled logic bits 1 and 0 for rescue at *h* and *i* in V_2 back to the original free bits 0 and 1 as in V_1, respectively. Compared with previous methods [12, 13], this new step (⑤) enhances the fault detection capability of the resulting test vector V_3, thus helping in further reducing test data inflation caused by risky path elimination.

It is clear that the fully-specified test vector V_3 obtained from *P-II (Risky Path Elimination)* consists of three types of logic bits: (1) ***detection bits*** (*a*, *f*, *g*) for primary and secondary fault detection, (2) ***free bits*** (*e*, *h*, *i*) for fortuitous fault detection, and (3) ***risky-path-impact bits*** (*b*, *c*, *d*) that are *X*-filled bits for the successful rescue of risky path lsp_1. Note that V_3 has fewer free bits than V_1.



Conventional
P-II (Risky Path Elimination)
Test Cube Generation w/ Dynamic Compaction
Detection-Oriented X-Filling (Random-Fill, etc.)
X-Restoration for Risky Paths
X-Filling for Reducing Local LSA
Remaining Risky Path Masking
Non-Impact Change Bit Release
Detection Bits
Free Bits
Risky-Path-Impact X-Bits
Risky-Path-Impact Bits
Risky
Safe
Impact Areas
lsp1
lsp2
P-I (Risky Path Checking)

Fig. 6. Example of risky path elimination.

3.3 Clock Stretch Reduction

Clock Stretch Reduction (*P-III*: ⑥~⑧ in Fig. 2) is to reduce *launch switching activity* in the impact areas of clock paths so as to reduce their test clock stretch. The first step is to select clock paths to be targeted. This is because not all clock paths have the same impact on at-speed test quality in terms of test clock stretch. For example, the clock stretch of the clock path for the endpoint FF of an ***unsensitized*** logic path has no adverse impact on at-speed test quality, while the clock stretch of the clock path for the endpoint FF of a sensitized logic path is less severe if the path is short. Therefore, the sensitization status as well as the length of a logic path need to be considered in order to determine whether its corresponding clock path needs to be targeted in clock stretch reduction or not.

Definition 4: A ***target clock path*** of a test vector *V* is the clock path for the endpoint FF of a *long sensitized* (logic) *path* (*LSP*) of the test vector *V*.

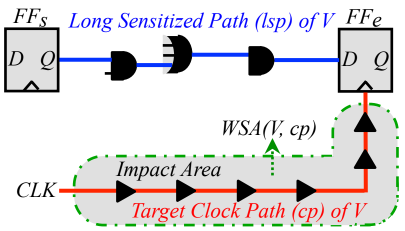
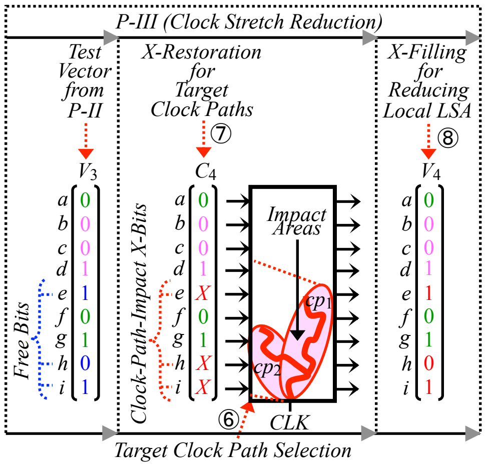


Fig. 7. Target clock path.

Fig. 7 illustrates a target clock path *cp* of a test vector *V*, corresponding to a long (logic) path *lsp* sensitized by *V*.



P-III (Clock Stretch Reduction)
Test Vector from P-II
X-Restoration for Target Clock Paths
X-Filling for Reducing Local LSA
Free Bits
Clock-Path-Impact X-Bits
Impact Areas
cp1
cp2
CLK
Target Clock Path Selection

Fig. 8. Example of clock stretch reduction.

P-III (*Clock Stretch Reduction*) is conducted as follows: First, ***target clock path selection*** (⑥) is conducted to identify all target clock paths of the test vector V_3, which is the resulting test vector from *P-II* (*Risky Path Elimination*). Assume that the identified target clock paths are cp_1, cp_2, …, and cp_n. Then, ***X-restoration for target clock paths*** (⑦) is conducted to (1) identify all free bits in V_3 that can reach the impact area of at least one target clock path and (2) turn those bits back into *X*-bits (referred to as *clock-path-impact X-bits*). The result is a new partially-specified test cube C_4. After that, ***X-filling for reducing local LSA*** (⑧) is conducted on C_4 to reduce *launch switching activity* (*LSA*) in the impact areas of the target clock paths. This results in a new fully-specified test vector V_4 with reduced local *LSA* around its target clock paths cp_1, cp_2, …, and cp_n, estimated by $WSAcp(V_4, cp_1)$, $WSAcp(V_4, cp_2)$, …, and $WSAcp(V_4, cp_n)$, respectively. As a result, the test clock stretch of V_4 can be reduced. As shown in Fig. 2, V_4 is the final test vector *V* for the current test generation run.

An illustrative example is shown in Fig. 8. Here, ***target clock path selection*** (⑥) identifies two target clock paths of the test vector V_3, namely, cp_1 and cp_2. Since all of the *free bits* (*e*, *h*, *i*) of V_3 can reach the impact areas of cp_1 and cp_2, all of them are turned into *clock-path-impact X-bits*. Then, 1, 0, and 1 are filled into the three *X*-bits at *e*, *h*, and *i*, respectively, for reducing the local LSA in the impact areas of the two target clock paths (cp_1 and cp_2).

4. Experimental Results

The proposed LCPA flow was implemented in *C* based on a commercial ATPG tool, and evaluated on large ITC'99 circuits with a workstation (CPU: Intel Xeon™ 3.33 GHz).

Table 1 shows the result of the baseline ATPG (A~E in Fig. 2 conducted by the commercial ATPG tool without the low-capture-power option), including basic circuit statistics (*# of Gates*, *# of FFs*, and *Max. Path Length*). Test costs are evaluated by test vector count (*# of Vectors*), while test quality is evaluated by transition delay fault coverage (*FC*), bridging coverage estimate (*BCE*) [14], and statistical delay quality level (*SDQL*) [3]. BCE and SDQL are for assessing the detection capability for bridging defects and small-delay defects, respectively. *P-I* (*Risky Path Checking*) was conducted, and the number of risky paths is shown under *# Risky Paths*. In addition, the average *launch switching activity* measured by WSA in the impact areas of clock paths for all test vectors is shown under *ave. WSAcp*. In the experiments, 70% of the maximum path length is used as the threshold for long paths, while the value of α in the new metric E3 for determining whether a long sensitized path is risky was set as 0.8. The results of the commercial ATPG with the low-capture-power option are also shown. It can be seen that using this option causes significant test data inflation w.r.t. the baseline ATPG and risky paths often still remain.

Table 2 shows the result of the proposed ATPG. The change rates of test vector count, transition delay fault coverage, BCE, and SDQL w.r.t. the baseline ATPG are shown under *Δ# of vectors(%)*, *ΔFC(%)*, *ΔBCE(%)*, and *ΔSDQL(%)*, respectively; the numbers of risky paths before and after *P-II* (*Risky Path Elimination*) are shown under *# Risky Paths (initial)* and *# Risky Paths (final)*, respectively; the change rate of the average WSA values for the impact areas of clock paths for all test vectors w.r.t. the baseline ATPG is shown under *Δave. WSAcp (%)*. Detailed information on *P-II* and *P-III* is provided under *P-II* and *P-III* in terms of the average percentage of risky-path-impact *X*-bits (*Ave.% RPI X-Bits*) at ②, the average success rate of turning risky paths into safe paths (*Ave. Rescue Rate*) by *X*-filling at ③, and the average number of masked response bits for remaining risky paths (*Ave.# of Masked Res. Bits*) at ④ for test vectors initially with risky paths, as well as the average percentage of clock-path-impact *X*-bits (*Ave.% CPI X-Bits*) at ⑦ for all test vectors.

Observations:

(1) The proposed LCPA scheme avoids any false capture failures in a guaranteed manner. This is because, if *X*-filling (③ in Fig. 2) cannot sufficiently reduce launch switching activity around a risky path, its endpoint is then masked in the corresponding test response data. Note that this masking (④ in Fig. 2) only modifies test data and thus incurs no hardware overhead.

(2) The proposed LCPA scheme significantly reduces launch switching activity around clock paths of long sensitized logic paths. This is the first work that has demonstrated that a test generation approach can indeed reduce test clock stretch effectively. Compared with the previous calibration-based solution [10], the proposed scheme accurately reduces test clock stretch on a per-FF/per-vector basis without any circuit design change and hardware overhead.

(3) The proposed LCPA scheme has a very low overhead in terms of small test data inflation (e.g., 3.6% for the LCPA scheme and 182.3% for the commercial ATPG tool in the case of b19). Note that the commercial ATPG tool only grossly reduces capture power without guaranteeing capture power safety (i.e., risky paths may remain), as evidenced in Table 1. In addition, it cannot mitigate the problem of test clock stretch.

(4) The basic LCPA concept is applicable to low-capture-power test compression [15]. Especially, it can be readily extended to broadcaster-based test compression through simple circuit model extension [16].

5. Conclusions

This paper has proposed the first *LCPA* (*Logic/Clock-Path-Aware*) at-speed scan test generation scheme for mitigating the impact of excessive launch switching activity in capture mode on logic paths (*in terms of false capture failures*) and on clock paths (*in terms of test clock stretch*). The LCPA scheme can more accurately and efficiently achieve capture power safety with very low test data inflation. In addition, this work has demonstrated for the first time that test clock stretch can be effectively reduced with a novel test generation technique.

Future work includes (1) more accurate simulation (e.g., timing-and-glitch-aware) and (2) a systematic approach to setting thresholds in risky path checking.

Acknowledgements

This work was jointly supported by JSPS Grant-in-Aid for Scientific Research (B) #25280016, JSPS Grant-in-Aid for Scientific Research on Innovative Areas #15K12003, JSPS Grant-in-Aid for the Promotions of Bilateral Joint Research Projects (Japan-Germany), the German Academic Exchange Service DAAD (BMBF), and the DFG under grant WU 245/16-1 (PARSIVAL).

References

[1] D. M. Walker, et al, *Delay Testing*, in *System-on-Chip Test Architectures*, Morgan Kaufmann, 2007.

[2] T. Mclaurin "Creating Structural Patterns for At-Speed Testing: A Case Study," *IEEE Design & Test of Computers*, Vol. 30, No. 2, pp. 66-76, Mar.-Apr. 2013.

[3] Y. Sato, et al., "Invisible Delay Quality - SDQM Model Lights Up What Could Not Be Seen," *Proc. Intl. Test Conf.*, Paper 47.1, 2005.

[4] X. Lin, et al., "Timing-Aware ATPG for High Quality At-Speed Testing of Small Delay Defects," *Proc. Asian Test Symp.*, pp.139-146, 2006.

[5] J. Saxena, et al., "A Case Study of IR-Drop in Structured At-Speed Testing," *Proc. Intl. Test Conf.*, pp. 1098-1104, 2003.

[6] X. Wen, et al., "On Low-Capture-Power Test Generation for Scan Testing," *Proc. IEEE VLSI Test Symp.*, pp. 265-270, 2005.

[7] S. Ravi, "Power-Aware Test: Challenges and Solutions," *Proc. Intl. Test Conf.*, Lecture 2.2, 2007.

[8] P. Girard, et al., Eds., *Power-Aware Testing and Test Strategies for Low Power Devices*, Springer, New York, 2009.

[9] M. Tehranipoor, et al, "Power Supply Noise: A Survey on Effects and Research," *IEEE Design & Test of Computers*, Vol. 27, No. 2, pp. 51-67, Mar.-Apr. 2010.

[10] J. Rearick, et al, "Calibrating Clock Stretch during AC Scan Test," *Proc. Int'l Test Conf.*, Paper 11.3, 2005.

[11] R. Franch, et al, "On-Chip Timing Uncertainty Measurements on IBM Microprocessors," *Proc. Int'l Test Conf.*, Paper 1.1, 2008.

[12] X. Wen, et al., "Power-Aware Test Generation with Guaranteed Launch Safety for At-Speed Scan Testing", *Proc. VLSI Test Symp.*, pp. 166-171, 2011.

[13] X. Wen, et al., “On Pinpoint Capture Power Management in At-Speed Scan Test Generation”, Proc. *Int'l Test Conf.*, Paper 6.1, 2012.

[14] B. Benware, et al., “Impact of Multiple-Detect Test Patterns on Product Quality,” *Proc. Intl. Test Conf.*, pp. 1031-1040, 2003.

[15] J. Rajski, et al., “Low Power Compression Utilizing Clock-Gating,” *Proc. Intl. Test Conf.*, Paper 7.1, 2011.

[16] K. Enokimoto, et al., “On Guaranteeing Capture Safety in At-Speed Scan Testing With Broadcast-Scan-Based Test Compression”, *Proc. 26th Intl. Conf. on VLSI Design*, pp. 279-284, 2013.

Table 1. Results of the Conventional ATPG

Circuit	*# of Gates*	*# of FFs*	*Max. Path Length*	*Baseline ATPG (w/o LCP)*							*ATPG (w/ LCP)*	
				# of Vectors	*FC (%)*	*BCE (%)*	*SDQL*	*# Risky Paths*	*Ave. WSAcp*	*CPU (Sec.)*	*Δ# of Vectors (%)*	*# Risky Paths*
b17	32326	1415	34	1175	70.1	67.2	16.3	31	167	674	141.2	68
b18	114621	3320	47	2428	63.6	64.8	198.9	172	400	5901	116.2	572
b19	231320	6642	68	3327	64.6	64.9	290.0	230	425	8175	**182.3**	809
b20	20226	490	45	1896	92.9	59.2	115.1	0	420	169	59.4	2
b21	20571	490	44	1870	93.2	58.9	134.7	0	573	139	55.5	4
b22	29951	735	50	2303	93.7	63.8	139.4	83	644	483	39.3	120

Table 2. Results of the Proposed ATPG

Circuit	*Proposed ATPG*											
	P-II			*P-III*								
	Ave. % of RPI X-Bits	*Ave. Rescue Rate (%)*	*Ave. % of Masked Res. Bits*	*Ave. % of CPI X-Bits*	*Δ# of Vectors (%)*	*ΔFC (%)*	*ΔBCE (%)*	*ΔSDQL (%)*	*# Risky Paths (initial)*	*# Risky Paths (final)*	*ΔAve. WSAcp (%)*	*CPU (Sec.)*
b17	32.3	0	78.9	45.6	4.7	0.1	0.1	-0.9	38	0	-8.4	982
b18	17.9	0	37.0	53.7	4.2	0.0	0.5	-0.1	335	0	-23.8	7078
b19	32.2	15.9	15.6	22.5	**3.6**	0.0	0.3	0.3	149	0	-14.8	14079
b20	N/A	N/A	N/A	39.5	8.3	0.0	1.2	-0.1	0	0	-15.0	271
b21	N/A	N/A	N/A	35.8	4.9	0.0	1.2	-0.5	0	0	-0.9	149
b22	16.1	0	25.7	36.1	12.6	0.0	1.2	-1.0	210	0	-18.5	379
Ave.	24.6	4.0	39.3	38.8	6.4	0.0	0.8	-0.4	122	0	-13.6	