PAPER

Testable Critical Path Selection Considering Process Variation

Xiang FU[†a)], ***Nonmember***, **Huawei LI**[†b)], ***Member***, ***and*** **Xiaowei LI**[†c)], ***Nonmember***

SUMMARY Critical path selection is very important in delay testing. Critical paths found by conventional static timing analysis (STA) tools are inadequate to represent the real timing of the circuit, since neither the testability of paths nor the statistical variation of cell delays caused by process variation is considered. This paper proposed a novel path selection method considering process variation. The circuit is firstly simplified by eliminating non-critical edges under statistical timing model, and then divided into sub-circuits, while each sub-circuit has only one prime input (PI) and one prime output (PO). Critical paths are selected only in critical sub-circuits. The concept of partially critical edges (PCEs) and completely critical edges (CCEs) are introduced to speed up the path selection procedure. Two path selection strategies are also presented to search for a testable critical path set to cover all the critical edges. The experimental results showed that the proposed circuit division approach is efficient in path number reduction, and PCEs and CCEs play an important role as a guideline during path selection.

key words: *testable critical path selection, process variation*

1. Introduction

Static timing analysis (STA) had been widely adopted in commercial EDA tools before the 90-nm era. However, as the dimension size of devices continues to shrink, the performance of devices becomes more and more vulnerable to process variation [1]. Because of the lack of predictability on delay variation of semiconductor devices, the conventional STA tools fail to find a complete critical path set that could represent the real timing of the circuit, or the found critical path set has poor testability.

Researchers have been driven to investigate appropriate model to tackle the statistical variation. In [2], the non-systematic process variation is classified into two classes, within-die variation and die-to-die variation. The within-die variation further contains two categories, spatially correlated and independent. Several research works have been conducted to develop proper models to handle both gate delay variations [3] and interconnection delay variations [4].

Confronting the problem of process variation, statistical STA (SSTA) is a possible solution. SSTA, which is in contrast with traditional deterministic STA (DSTA), adopts the delay variation information of gates and interconnections. Early work of SSTA is proposed in [5]. These research works focus on path-based approaches [5], [6]. However, the countless paths in modern ASIC circuits hindered the development of such approaches. Recent years witness the maturation of SSTA, and block-based SSTA becomes popular. The block-based methods follow the DSTA algorithm more closely. The arrival time at each node is computed by traversing the circuit graph in a topological manner. It's a linear approach and efficient in CPU time. Works on block-based SSTA approaches can be found in [7], [8]. These papers mainly concentrate on the propagation of mean delay and delay deviation of cells.

The testability of paths should also be considered when selecting critical paths. As we know, the total number of paths is exponential to the size of a circuit, while most of the paths are robustly untestable, or even non-robustly untestable, as reported by previous automatic test pattern generation (ATPG) research in delay testing [9]. In previous works, the testability of paths has been considered in both crosstalk induced delay testing [10] and small delay defect testing [11], [12]. In [13], an approach to reduce the number of paths to be tested is proposed, even though the final path set is reduced significantly, the rigorous test on short paths will result in over-testing.

This paper focuses on finding the testable critical paths considering process variation. A critical path, in this paper, is defined as a path which has the potential to be the longest path under certain conditions caused by process variation. The proposed testable critical path selection method includes two procedures.

In the first procedure, the circuit under test (CUT) is simplified by eliminating non-critical edges at the logic level, and then divided into sub-circuits, while each sub-circuit has only one prime input (PI) and one prime output (PO). The maximum delay of each sub-circuit is estimated using a statistical timing model at the logic level. If the estimated maximum delay has the potential to be larger than the original circuit's maximum delay obtained by STA, this sub-circuit is regarded as critical and should be retained for critical path selection. By sub-circuit partition, the concept of partially critical edges (PCEs) and completely critical edges (CCEs) are introduced, and critical paths considering process variation can be efficiently identified.

In the second procedure, non-robust delay test generation is done on selected paths in critical sub-circuits, and two path selection strategies are presented to find the least

Manuscript received June 30, 2009.
Manuscript revised September 3, 2009.

[†]The authors are with the Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academic of Science, Beijing, 100190, China, and with the Graduate University of Chinese Academic of Sciences, Beijing, 100039, China.

a) E-mail: fuxiang@ict.ac.cn
b) E-mail: lihuawei@ict.ac.cn
c) E-mail: lxw@ict.ac.cn

DOI: 10.1587/transinf.E93.D.59

number of testable paths that cover as many critical edges as possible.

Experimental results on ISCAS'89 benchmark circuits showed that the proposed circuit division approach is efficient in path number reduction, and PCEs and CCEs play an important role as a guideline during path selection.

The rest of the paper is organized as follows. Section 2 provides the motivation of this work. In Sect. 3, the proposed SSTA flow on critical path analysis is introduced. Section 4 describes the two testable path selection strategies. Experimental results on ISCAS'89 benchmark circuits are shown in Sect. 5. Finally, we present our conclusions in Sect. 6.

2. Motivation

The conventional DSTA approach is based on the classic critical path identification algorithm. For a circuit with fixed timing, the signal *latest arrival time* and the *minimum required time* (definition 1 and definition 2) of an edge can be calculated by one time traverse and back traverse of the circuit graph.

Definition 1: The signal *latest arrival time* of an edge is defined as the longest sub-path delay starting from any PI(PPI) to the edge, and denoted as $T_{arrival\_latest}$.

Definition 2: The signal *minimum required time* of an edge is defined as the longest path delay of the whole circuit minus the longest sub-path delay starting from the edge to any PO(PPO), and denoted as T_{req_min}.

Based on the above definition, the paths constructed by the edges which have equal $T_{arrival\_latest}$ and T_{req_min} are the longest paths of the circuit with fixed timing. The criterion is simple and efficient. This makes the conventional STA tools very good at finding the longest path in fixed timing mode. However, due to process variation, the timing of fabricated circuits varies from one to another. So paths which are not the longest in one circuit may be the longest in another one, and may escape from the critical path set if fixed timing is used in STA. An example of conventional DSTA is displayed in Fig. 1. In Fig. 1, the edge delay, signal *latest arrival time* and *minimum required time* are displayed in the format of *delay*, $<T_{arrival\_latest}, T_{req\_min}>$.

Given a slack threshold *slack_thsld*, if the $T_{arrival\_latest}$ and T_{req_min} of an edge do not satisfy Eq. (1), the edge is a non-critical edge, and all the paths through the edge are not critical. This kind of edges can be eliminated, and a *circuit skeleton* solely constructed by critical edges is extracted.

$$T_{req\_min} - T_{arrival\_latest} \leq slack\_thsld \quad (1)$$

If we want to find paths each of which has a slack not larger than 1, let the *slack_thsld* of the circuit in Fig. 1 be 1, then there are three edges and two nodes (highlighted in red) can be removed. However, after all of the non-critical edges are eliminated, the *circuit skeleton* (constructed by the black arches and nodes) still has short paths. For instance, path *PI2-A1-A4-A7-PO*, whose on-path edges all satisfy Eq. (1), is not a critical path. The on-path edges of path *PI2-A1-A4-A7-PO* such as *A1-A4* and *A4-A7* are different from edges such as *A3-A4* and *A4-A6*. Apparently, the definition of $T_{arrival\_latest}$ and T_{req_min} cannot tell the difference between these two kinds of edges. In this paper, edges like *A1-A4* and *A4-A7* are defined as partially critical edges (PCEs), because there are short paths through this kind of edges, but not all. Edges like *A3-A4* and *A4-A6* are defined as completely critical edges (CCEs), through which all paths are critical. The identification of PCEs and CCEs is discussed in the next section.

Exhaustive enumeration of the whole path set will no doubt find all the long paths. However, it's timing consuming or even impossible to enumerate the whole path set. On the other hand, the CPU efficiency of conventional DSTA is an advantage that we cannot resist. Divide and conquer is a good solution for this problem. If a non-critical edge is eliminated, all paths through the edge needn't to be considered in critical path selection. We observed through experiments that the number of non-critical edge is usually pretty high, and lots of paths can be excluded at the early stage. The *circuit skeleton* can be further divided into the sub-circuits according to the PI(PPI)/PO(PPO) pairs. The sub-circuit, which is also named as the logic cone of the PI(PPI)/PO(PPO) pair in this paper, becomes so simple that enumerating its whole path set becomes possible. What's more, one can reduce a given subset of paths effectively by removing paths that have non-critical edges, which meets the need of practice in timing verification and delay testing.

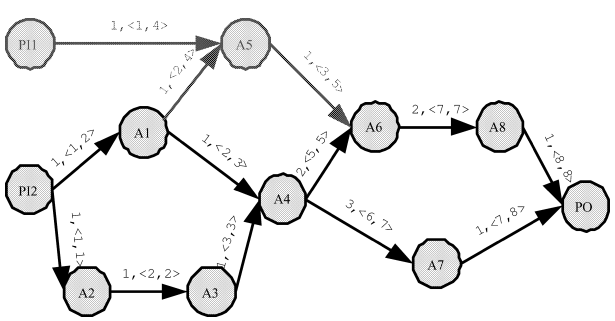


Fig. 1 An example of conventional DSTA.

3. The Proposed SSTA Flow

The proposed SSTA flow is shown in Fig. 2. Firstly, the SSTA for the whole circuit is performed. Secondly, an early stage path reduction scheme by eliminating the non-critical edges and a circuit partition procedure which divides the *circuit skeleton* into sub-circuits between the PI(PPI) and PO(PPO) pairs are performed. Thirdly, an SSTA procedure is further executed in each cone. This SSTA procedure identifies whether the extracted logic cone is critical or not. If it is not critical, the next PI(PPI)/PO(PPO) pair in the *circuit skeleton* is selected; else, the PCEs and CCEs in the logic cone are identified, and the path selection procedure is performed.

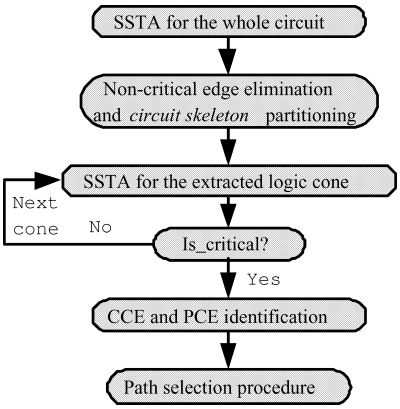


Fig. 2 The proposed SSTA flow.

Table 1 Delay variations of basic gates.

Gate Type	Mean(μ)	Standard Deviation(σ)
BUFF	4	0.3
INV	2	0.2
NOR,NAND	3	0.3
OR,AND	4	0.3

3.1 SSTA Timing Model and Path Selection Criterion

In the proposed method, block-based SSTA is adopted. For simplicity, only independent within-die variation is considered. This means the delay variations of gates (also called as timing edges), are assumed to be independent to each other. If two paths have a big ratio of common part, the delay distribution of the two paths will be probably dependent on each other. Several techniques take the spatial correlation into consideration [14]. It's more accurate than the SSTA methods that regard the timing edges as independent.

The delay variation of a cell is assumed to obey Gauss distribution, and is denoted as (μ, σ), where μ is the mean value of the cell delay, and σ is the delay standard deviation of the cell delay. In the experiments of this paper, the mean delay and delay standard deviation of each type of basic logic gate are educed from the approximate normalized delay value of a 0.18-um library, and are listed in Table 1. Although the propagation delay of a rising transition is different from that of a falling transition in practice, the proposed SSTA doesn't take the difference into account for simplicity. As a result, the limitation of the proposed SSTA method is the lack of accuracy. However, we must add that the strength of our path selection method is not lessened by using the simple delay model. One can use other more accurate methods to model the delay and delay variations at the logic level, which is not the focus of this paper.

As we can see in Table 1, the standard deviation σ of a certain gate is assumed to be around 10% of the normal delay value. Thus, 3σ equals 30% of the normal delay value. It should be noted that, as the number of on-path gates increases, the variation of the path delay decreases since paths with more gates are less sensitive to process variation. In the experiment of the proposed method, a 5–10% path delay variation is obtained.

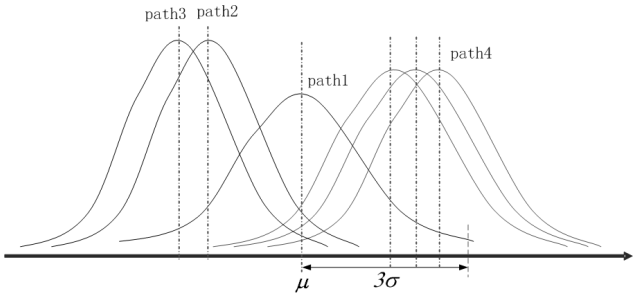


Fig. 3 Probability distribution function of the paths.

Since the delay variations of on-path gates of a path are assumed to be independent to each other, the SSTA could treat the delay of a path, which is total delay of the on-path gates, with an approximation of Gaussian distribution. The mean delay μ_p, and delay standard deviation σ_p, of a path p with n gates can be calculated by Eqs. (2) and (3).

$$\mu_p = \sum_{i=1}^{n} \mu_i \tag{2}$$

$$\sigma_p = \sqrt{\sum_{i=1}^{n} \sigma_i^2} \tag{3}$$

where μ_i, σ_i $(i = 1, \ldots, n)$ are the mean delay and delay standard deviation of the *i-th* on-path gate respectively. The probability distribution function (PDF) of the path delays can be simply illustrated in Fig. 3. The paths highlighted in red are the critical paths identified by the conventional STA tools. *path1* has a mean delay of μ, and the corresponding delay standard deviation is σ.

The mean delay μ of *path1* plus 3σ exceeds the mean delay of a critical path, *path4*. So *path1* has the potential to be the longest path, and should be considered. The path selection criterion of critical paths can be further described as follows: If the delay of a path p satisfies Eq. (4), the path is critical.

$$\mu_p + 3\sigma_p \geq \mu_{max\_circuit} \tag{4}$$

where $\mu_{max\_circuit}$ is the maximum mean delay of the circuit, which is equal to the maximum delay calculated by STA.

In this paper, for simplicity, we assume that paths sharing the same PI(PPI) and PO(PPO) have a unified σ_p value which equals the delay standard deviation of the longest path between the PI(PPI)/PO(PPO) pair (denoted as σ_{cone}).

3.2 Path Number Reduction and Circuit Partition

Path reduction in early stage is of great importance. If the $T_{arrival\_latest}$ and T_{req_min} of an edge do not satisfy Eq. (5), the edge is a non-critical edge.

$$T_{req\_min} - T_{arrival\_latest} \leq 3\sigma_{max\_circuit} \tag{5}$$

where, $\sigma_{max\_circuit}$ is the maximum delay standard deviation of the circuit. The non-critical edges can be eliminated, and a *circuit skeleton*, which solely constructed by critical edges, is extracted. The PI(PPI)s and the PO(PPO)s in the *circuit skeleton* are defined as *critical PI(PPI)s* and *critical PO(PPO)s*.

To further reduce the path set, the *circuit skeleton* can be divided into path groups which share the same *critical PI(PPI)* and *critical PO(PPO)*. For a certain critical PI(PPI)/PO(PPO) pair, the logic cone between the PI(PPI)/PO(PPO) pair is the intersection of the fan-out cone of the PI(PPI) and the fan-in cone of the PO(PPO).

If a PI(PPI) or a PO(PPO) is not a critical one, no path through the PI(PPI) or PO(PPO) is critical. But this doesn't mean there must be a critical path through a critical PI(PPI) or critical PO(PPO) because the slack threshold used in Eq. (5) is the maximum delay deviation of the whole circuit ($3\sigma_{max\_circuit}$). It's a conservative value. In the proposed SSTA flow (Fig. 2), to further identify whether there exists a critical path in the logic cone of each PI(PPI)/PO(PPO) pair, an SSTA procedure is performed exclusively on the logic cone while both the PI(PPI) and the PO(PPO) are critical. Let the delay standard deviation of the longest path in the cone be σ_{cone}, and the maximum mean delay of the cone be μ_{max_cone}. If σ_{cone} and μ_{max_cone} satisfy Eq. (6), the logic cone is a critical one. Only when a logic cone is critical, is the path selection procedure in the cone invoked.

$$\mu_{max\_cone} + 3\sigma_{cone} \geq \mu_{max\_circuit} \tag{6}$$

3.3 Identifications of PCEs and CCEs

Though in a critical logic cone, there may exist paths that are not critical. Since the information of shorter paths is lost during the calculation of $T_{arrival\_latest}$ and T_{req_min}, the conventional DSTA can not exclude shorter paths through critical edges efficiently. From the definitions of $T_{arrival\_latest}$ and T_{req_min} of an edge, we can know that the slack of the longest path through the edge, which is also the minimum slack of all paths through the edge, can be calculated by Eq. (7).

$$slack\_min = T_{req\_min} - T_{arrival\_latest} \tag{7}$$

In order to get the maximum slack of all paths through the edge, the definitions of signal *earliest arrival time* and *maximum required time* are introduced.

Definition 3: The signal *earliest arrival time* of an edge is defined as the shortest delay starting from any PI(PPI) to the edge, and denoted as $T_{arrival\_earlist}$.

Definition 4: The signal *maximum required time* of an edge is defined as the longest path delay of the whole circuit minus the shortest delay starting from the edge to any PO(PPO), and denoted as T_{req_max}.

Since $T_{arrival\_earlist}$ and T_{req_max} are affected by the shorter paths, $T_{arrival\_earlist}$ and T_{req_max} are calculated on the extracted *circuit skeleton*, without considering the non-critical edges. The slack of the shortest path through a certain edge, which is also the maximum slack of all the paths through the edge, can be calculated by Eq. (8). Thus, the slacks of all paths though the edge belong to a span from *slack_min* to *slack_max*.

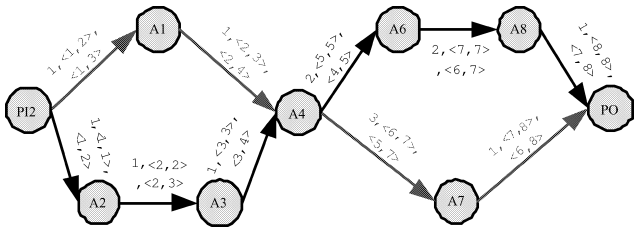


Fig. 4 CCEs and PCEs of circuit in Fig. 1.

$$slack\_max = T_{req\_max} - T_{arrival\_earliest} \tag{8}$$

Given a slack threshold *slack_thsld*, the following conditions should be considered.

$slack\_max < slack\_thsld$ (condition 1)
$slack\_min \leq slack\_thsld \leq slack\_max$ (condition 2)
$slack\_min > slack\_thsld$ (condition 3)

In *condition 1*, the *slack_max* is smaller than *slack_thsld*. This means that all of the paths through this edge are critical, and such edge is considered as completely critical edge (CCE). In *condition 2*, *slack_min* is not larger than *slack_thsld*, while *slack_max* is not smaller than *slack_thsld*. This means that there are paths through the edge are not critical, but not all. Such edge is defined as partially critical edge (PCE). In *condition 3*, *slack_min* is larger than *slack_thsld*, so no path through the edge is critical. Such edge is defined as non-critical edge (NE).

The circuit in Fig. 4 is the *circuit skeleton* of the circuit in Fig. 1 after the process of critical PI(PPO)/PO(PPO) identification and non-critical edge elimination. The $T_{arrival\_earlist}$ and T_{req_max} are displayed below the $T_{arrival\_latest}$ and T_{req_min}. Let *slack_thsld* be 1. In Fig. 4, the PCEs are the edges highlighted in red, while the CCEs are the black edges. In the experiments of this paper, to identify the PCEs and CCEs in the *circuit skeleton*, *slack_thsld* is specified to $3\sigma_{max\_circuit}$.

However, identification of the CCEs and PCEs in the critical logic cones is different from what it is in the *circuit skeleton*. As mentioned above, the *circuit skeleton* is further partitioned into several logic cones. Each logic cone has only one PI(PPI) and only one PO(PPO), and an SSTA is performed on it exclusively. During this phase, the $T_{req\_max\_cone}$, $T_{req\_min\_cone}$, $T_{arrival\_latest\_cone}$, $T_{arrival\_earlist\_cone}$ of each edge in the cone are re-calculated, without considering the edges that don't belong to the current logic cone. The *slack_min_cone* and *slack_max_cone* of the edges are calculated according to Eqs. (7) and (8), and the *slack_thsld_cone* is specified to $3\sigma_{cone}$ in stead of $3\sigma_{max\_circuit}$.

As a result, according to the path selection criterion (Eq. (4)), if a path contains at least one CCE, the path is definitely a critical path. However, if a path is constructed solely by PCEs, the path may be a critical path, or may not.

Thus, one more justification procedure is needed to justify whether the mean delay of the path satisfies the critical path selection criterion. Therefore, uncovered CCEs gain higher priority during path selection to reduce the requirement of justification procedures.

As we can see, the identification of CCEs and PCEs in the logic cones is more accurate than the identification in the *circuit skeleton*. Since there is only one PI(PPI) and only one PO(PPO) in the logic cone, the longest (or shortest) path through an edge in the logic cone is not necessarily longer (or shorter) than the longest (or shortest) path through the same edge in the *circuit skeleton*. The *slack_min_cone* (or *slack_max_cone*) of the edge in the logic cone is therefore larger (or smaller) than or equal to, the *slack_min* (or *slack_max*) of the same edge in the *circuit skeleton*. The slack span of the edge in the logic cone, which is from *slack_min_cone* to *slack_max_cone*, tends to be narrower than the slack span in the *circuit skeleton*, which is from *slack_min* to *slack_max*. The result of a narrower slack span is that there will be fewer edges satisfying *condition 2* in the logic cones and more edges satisfying *condition 1* or *condition 3*. In other words, the PCE ratio of a logic cone will be lower than the PCE ratio of the *circuit skeleton*. The experimental results of this paper validate the above analysis, while the PCE ratio of *circuit skeleton* is much higher than the average PCE ratio of the logic cones. High ratio of PCEs will call for lots of critical path justification procedures, and will lower the efficiency of path selection procedure.

4. Path Selection and ATPG

As mentioned before, in a critical logic cone, there may exist paths that are not critical. So after a critical logic cone is identified, we can further get a total, testable and critical path set in the cone, for timing analysis during design, or for delay testing. For the purpose of test cost reduction, it is also desirable to find a minimum testable critical path set to cover all the PCEs and CCEs, or cover the PCEs and CCEs as many as possible. A combinational ATPG is performed to justify whether the selected path is testable. In the ATPG procedure, non-robust path sensitization criterion [15] is utilized. For a physical path, there are two logical path delay faults (PDFs). The two PDFs correspond to the propagation of a rising transition and a falling transition at the path input respectively. If not specified, a path refers to a physical path in this paper. A path is said to be testable only when both of two logical PDFs are testable.

In this section, two path selection strategies are proposed. The first one is a greedy strategy, and the second one is a pseudo exhaustive strategy. The two strategies are appropriate to different kinds of circuits.

4.1 Greedy Path Selection Strategy

As it is shown in the SSTA flow, the path selection procedure is only executed on the logic cones which have critical paths. The logic cones are the simplified sub-circuits that only contain CCEs and PCEs. In the path selection procedure, these CCEs and PCEs are marked as uncovered edges initially, and stored according to their topological order. The path selection flow of the greedy approach is shown in Fig. 5.

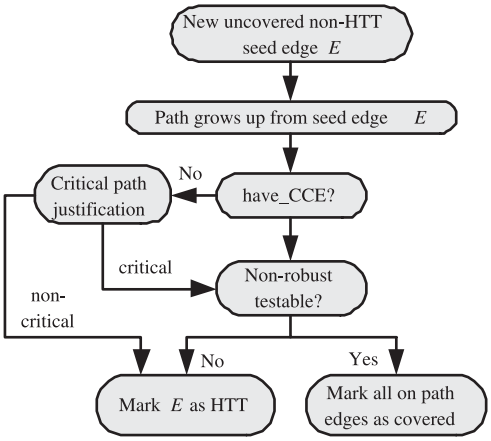


Fig. 5 Path selection flow of the greedy strategy.

Table 2 Priority lookup table in greedy path selection strategy.

Edge Type	Priority	
	Have_CCE=0	Have_CCE=1
Uncovered CCE	1	2
Covered CCE	2	4
Uncovered PCE	3	1
Covered PCE	4	3

For each uncovered edge, it is treated as a seed edge firstly. The path grows up from the seed edge by choosing edges from its fan-out edges and fan-in edges till the PI(PPI) and PO(PPO) are reached. If an edge with multiple fan-outs (or fan-ins) is encountered, the path selection procedure needs to judge which edge should be selected. A priority lookup table is established to help the edge selection, as shown in Table 2.

Since only one CCE is enough to assert that the selected path is critical, the priority of uncovered CCEs is higher if no CCE has been selected. But when a CCE has been selected, the uncovered PCEs take a priority over the others. In this way, the greedy strategy can find a path set with minimum path number to cover all of the critical edges. If there is no CCE in the selected path, a critical justification procedure is performed to justify the selected path is critical or not. If the path is not critical, the corresponding edge is marked as a hard-to-test (HTT) edge. In the proposed flow, the ATPG procedure is performed to justify whether the selected path is testable or not. If the selected path is not testable, the seed edge is marked as an HTT edge too. If the selected path is critical and testable, all the on-path edges are marked as covered edges.

The greedy strategy is efficient when the PCE ratio is not too high and most of the paths are testable. However, when the PCE ratio is high and the testability is low, most of the edges are marked as HTT edges, even though there

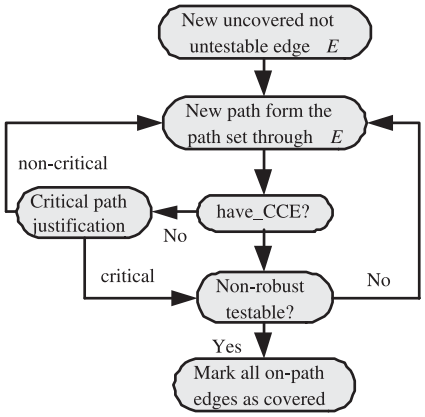


Fig. 6 Path selection flow of the pseudo exhaustive strategy.

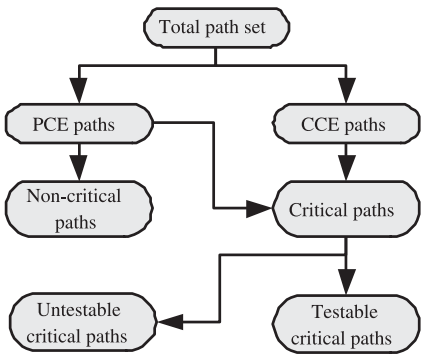


Fig. 7 Paths classification of the circuit.

are testable critical paths through these edges. To solve this problem, a pseudo exhaustive strategy is developed.

4.2 Pseudo Exhaustive Path Selection Strategy

The main defect of the greedy strategy is that it cannot assert whether all paths through a certain edge are untestable. It stops when a certain edge is marked as an HTT edge. But actually, for some HTT edges, there may exist paths that are critical and testable through them. Exhaustive search will help us to find out these paths.

In contrast with the greedy strategy, the pseudo exhaustive strategy flow (Fig. 6) stops till a testable critical path though a certain edge is found. If no path is found to be critical and testable, the edge can be marked as untestable. The untestability of edges can be propagated. If all the fan-in edges of a certain edge are marked as untestable, the edge can also be marked as untestable. This property can help to accelerate this strategy.

Unlike the greedy strategy, the pseudo exhaustive strategy needs a fan-in stack and fan-out stack to store the current path information. It is inflexible for the pseudo exhaustive strategy to choose fan-out (or fan-in) edges in a way like the greedy strategy does. But in the pseudo exhaustive strategy, the CCEs have a higher priority to be pushed into the stack when an edge with multiple fan-outs (or fan-ins) is encountered. This makes the paths first found by the pseudo exhaustive strategy are the paths containing CCEs. As a result, there is no need for critical path justification at the beginning. Apparently, the pseudo exhaustive strategy guarantees to find the maximum covered edge set, while the greedy strategy fails to do this.

The paths in a critical logic cone can be classified in a way as shown in Fig. 7. The PCE paths are the paths solely constructed by PCEs, while the CCE paths are the paths that contain at least one CCE. The untestable critical paths play the same role during path selection as the non-critical paths in the critical logic cone do. If a critical path is found to be untestable, the path needs to be re-selected or the corresponding seed edge is marked as an HTT edge.

5. Experimental Results

The proposed path selection strategy has been implemented in Visual C++ and run on Windows XP at a 2.6 GHz Pentium 4 processor with 512 MB memory. In the proposed flow, SSTA is performed for the whole circuit at first. The main purpose of this step is to identify the critical PI(PPI)/PO(PPO)s and eliminate the non-critical edges. Since the computational complexity of the proposed SSTA approach is linear to the number of timing edges, the CPU time of this step is negligible in comparison with the path selection and ATPG procedure. After this step, the *circuit skeleton* is divided into logic cones sharing the same critical PI(PPI) and PO(PPO). For each critical logic cone, the PCEs and CCEs are identified. The circuit information of the *circuit skeleton* and the critical logic cones are listed in Table 3.

The first column of Table 3 shows the circuit name. In the two sub-columns of Column 2, the total path number and the critical path number finally recorded are listed. The total edge number is listed in Column 3. Column 4 and Column 5 show the information of the *circuit skeleton* and critical logic cones respectively. In the three sub-columns of the Column 4 and Column 5, the path number, the critical edge number and the PCE ratio are listed from left to right. Column 6 shows the critical logic cone number. The testable critical path number is showed in the last column, this number is obtained by exhaustive test generation on all the critical paths.

As we can see, because most of the edges are non-critical, the *circuit skeleton* is much simpler than the original one. From the path number in the *circuit skeleton*, we can see that a large percentage of paths in the original circuit are excluded from further path selection. On average, only 3% paths are remained after SSTA for the whole circuit. As the number of critical logic cones is small and the cone logic is simplified, enumeration of all paths of the logic cones becomes possible.

In Table 3, a clear trend can be seen that, the PCE ratio and the number of paths are reduced significantly if we divide the circuit into logic cones that share the same PI(PPI) and PO(PPO). Obviously, the significant reduction of the PCE ratio will contribute to the speed-up of the path selec-

Table 3 Critical information after the SSTA flow.

Circuit name	Path #		Total Edge #	circuit skeleton			Critical logic cones			Critical cone#	Testable critical path #
	Total	critical		path #	Edge #	PCE ratio	path #	Edge #	PCE ratio		
C880	8642	291	906	765	136	100%	392	136	36.9%	16	218
S208	145	4	218	4	29	0	4	29	0	2	4
S344	355	6	361	8	42	80.9%	6	42	0	3	4
S444	535	20	471	24	53	64.2%	20	53	0	10	6
S953	1156	17	1005	27	112	95.5%	17	112	0	12	17
S1196	3098	40	1128	58	130	42.3%	40	118	0	11	9
S1423	44726	68	1502	160	133	100%	101	133	78.8%	6	4
S5378	13542	192	5523	384	155	100%	384	155	100%	12	192
S38584	1,080,723	9168	40162	19845	479	100%	18285	479	66.7%	6	0

Table 4 Experimental results of the two proposed strategies.

Circuit name	Greedy Strategy						Pseudo Exhaustive Strategy					
	Total Selected path #	Critical Path #	CCE Path #	Testable Critical Path #	CPU time (s)	Coverage	Total Selected path #	Critical Path #	CCE Path #	Testable Critical Path #	CPU time (s)	Coverage
C880	45	28	24	21	0.36	97.0%	92	57	43	26	0.44	100%
S208	3	3	3	3	0.18	100%	3	3	3	3	0.17	100%
S344	6	6	6	4	0.22	97.6%	6	6	6	4	0.19	97.6%
S444	16	16	16	2	0.35	56.6%	18	18	18	4	0.25	64.2%
S953	11	11	11	11	0.31	100%	11	11	11	11	0.31	100%
S1196	23	23	23	6	0.66	64.4%	27	27	27	6	0.38	64.4%
S1423	52	34	14	1	0.98	72.1%	101	67	44	3	0.41	75.9%
S5378	63	23	0	23	0.57	96.8%	59	31	0	31	0.63	100%
S38584	623	451	106	0	62.5	0	18285	9168	384	0	390	0

tion procedure in logic cones.

Some circuits, such as C880, S1423 and S38584, have a 100% ratio of PCE if we don't cut the circuit down. A 100% ratio of PCE means, for any path selected from the *circuit skeleton*, a critical path justification procedure is needed additionally. If the justification fails, the path must be re-selected. On the other hand, for circuits like S208, S344, S444, S953 and S1196, the PCE ratio is 0 if we cut down the circuits in the proposed way. A zero ratio of PCE means any paths in the critical logic cone are critical. In this situation, the path set in the logic cones is the very critical path set finally recorded. However, S5378 is an exception, no PCE ratio reduction can be seen after the *circuit skeleton* is partitioned. This is because most of the edges in S5378 have both short and long paths through them. The controllability of S5378 is high, and most of the paths are robustly testable. Anyway, for most of the circuits, the information in Table 3 shows the efficiency of the circuit partition procedure from another aspect.

In this paper, two path selection strategies are proposed. The experimental results of the two strategies are listed in Table 4. As shown in Fig. 7, there are non-critical paths in a critical logic cone so long as there are PCEs in the logic cone. It is inevitable for the path selection procedure to encounter the non-critical paths or untestable critical paths, so the total selected paths during the path selection procedure can also be classified in the same way. For each strategy, the number of total selected paths, the number of the selected critical paths, the number of selected CCE paths, the final number of selected testable critical paths, the CPU processing time and the critical edge coverage of the selected testable critical paths are listed in each sub-columns from left to right.

For small circuits such as S208, S344 and S953, no difference can be seen between the proposed two strategies. But for the other circuits except S5378, the paths selected by the greedy strategy are fewer than the paths selected by pseudo exhaustive strategy. Take C880 for example, 4 edges are marked as HTT edges in the greedy strategy. In order to cover the 4 HTT edges, the pseudo exhaustive strategy selects 92 paths, while the greedy strategy only selects 45 paths. The same phenomenon can be found in S1423. Lots of extra paths are selected in order to cover several HTT edges. The reason for this is that such circuit has both low testability and a visible PCE ratio. It is hard to find critical and testable paths to cover these HTT edges. For S5378, the paths selected by the greedy strategy are more than the pseudo exhaustive strategy. This is caused by two reasons. First, for a certain critical edge, it may reside in more than one critical cones. As a result, it is used as a seed edge for several times in different logic cones till it is covered, and several paths through this edge are selected. Same situation also happens on S38584, when there are 479 critical edges, but 623 paths are selected by the greedy strategy. Second, the PCE ratio of S5378 is 100% though all paths in the candidate path set are testable. Some edges are marked as HTT edges due to the critical path justification failure and the greedy strategy fails to achieve 100% critical edge coverage.

The testable path justification rule, which requires both

of the two logical PDFs are testable, is too tight or pessimistic for circuits whose long paths have poor testability. For large circuits like S38584, all of the critical paths found by the proposed method fail to satisfy this rule. Since all the critical paths found in S38584 are untestable, the pseudo exhaustive strategy turns up to an enumeration strategy. Fortunately, our later experiments show that for parts of the critical paths of S38584 found by the proposed method, there is one testable PDF (ether rising or falling), and the other one is untestable. So the testable path justification rule can be relaxed when either PDF of the path is testable to improve test coverage. On the other hand, test coverage can also be improved considering second-longest paths by increasing the *slack_thsld* of the circuit.

CCEs play an important role during path selection. In the selected critical paths, the CCE path number is considerable high. In the critical paths found by each strategy, lots of paths are directly justified as critical because of the CCEs they contain. This validates the efficiency brought by the classification of PCEs and CCEs.

6. Conclusion

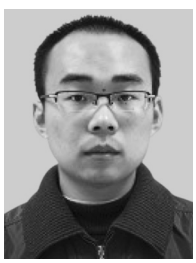
In this paper, a novel path selection method considering process variation is introduced. The non-critical edges are eliminated and a *circuit skeleton* solely constructed by critical edges is extracted. Then, the *circuit skeleton* is divided into logic cones between the PI(PPI)/PO(PPO) pairs. Critical logic cones are identified, and enumeration of paths in the cones becomes possible. The concept of PCEs and CCEs are introduced and used in the path selection strategy. The experimental results showed that the proposed circuit division approach is efficient in path number reduction, and PCEs and CCEs play an important role as a guideline during path selection.

Acknowledgements

This research is supported in part by National Natural Science Foundation of China (NSFC) under grant No.60776031, 60606008, 60633060, in part by Hi-Tech Research and Development Program of China (863) under grant No.2007AA01Z476, 2007AA01Z109 and 2007AA01Z113, and in part by National Basic Research Program of China (973) under grant No.2005CB321605.

References

[1] International Technology Roadmap for Semiconductor. http://www.itrs.net/Links/2008ITRS/Home2008.htm
[2] D. Blaauw, K. Chopra, A. Srivastava, and L. Schefter, "Statistical timing analysis: From basic principles to state of art," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.27, no.4, pp.589–607, April 2008.
[3] A. Agarwal, F. Dartu, and D. Blaauw, "Statistical gate delay model considering multiple input switching," Proc. DAC, pp.658–663, 2004.
[4] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula, "Variational delay metrics for interconnect timing analysis," Proc. DAC, pp.381–384, 2004.
[5] R. Brashear, N. Menezes, C. Oh, L. Pillage, and M. Mercer, "Predicting circuit performance using circuit-level statistical timing analysis," Proc. DATE, pp.332–337, 1994.
[6] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," Proc. DAC, pp.556–561, 2002.
[7] J. Le, X. Li, and L. Pileggi, "STAC: Statistical timing analysis with correlation," Proc. DAC, pp.343–348, 2004.
[8] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," Proc. ICCAD, pp.607–614, 2003.
[9] L.-C Wang, J.-J. Liou, and K.-T. Cheng, "Critical path selection for delay fault testing based upon a statistical timing model," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.23, no.11, pp.1550–1565, Nov. 2004.
[10] R. Tayade and J.A. Abraham, "Critical path selection for delay test considering coupling noise," Proc. ETS, pp.119–124, 2008.
[11] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern grading and pattern selection for small-delay defects," Proc. VTS, 2008.
[12] W. Qiu and J. Wang, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," Proc. ITC, pp.223–231, 2004.
[13] H. Li, Z. Li, and Y. Min, "Reduction of number of paths to be tested in delay testing," J. Electronic Testing: Theory and Applications, vol.16, no.5, pp.477–485, 2000.
[14] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," Proc. ICCAD, pp.900–907, 2003.
[15] 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.



Xiang Fu received B.S. degree in Hua Zhong University of Science and technology in 2005. He now is a Ph.D. candidate in computer science in Institute of Computing Technology, Chinese Academic of Science. His research interests include design for test and high quality delay testing.

Huawei Li received the B.S. degree in computer science from Xiangtan University, Hunan, China, in 1996, and the M.S. and Ph.D. degrees from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1999 and 2001, respectively. She is currently a Professor at the Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include VLSI/SOC design verification, test generation, delay test and design for reliability. She is a senior member of IEEE. She was a Technical Program Chair of IEEE Asian Test Symposium (ATS) in 2007, and Workshop of RTL and High Level Testing (WRTLT) in 2003.

Xiaowei Li received his B.Eng. and M.Eng. degrees in computer science from Hefei University of Technology (China) in 1985 and 1988 respectively, and his Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 1991. Dr. Li joined Peking University as a Postdoctoral Research Associate in 1991, and was promoted to Associate Professor in 1993, all with the Department of Computer Science and Technology. From 1997 to 1998, he was a Visiting Research Fellow in the Department of Electrical and Electronic Engineering at the University of Hong Kong. In 1999 and 2000, he was a Visiting Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. He Joined the Institute of Computing Technology, Chinese Academy of Sciences as a professor in 2000. His research interests include VLSI/Soc design verification and test generation, design for testability, low-power design, dependable computing. Dr. Li received the Natural Science Award from the Chinese Academy of Sciences in 1992, the Certificate of Appreciation from IEEE Computer Society in 2001. He is a senior member of IEEE. He is an area editor of the Journal of Computer Science and Technology and an associate editor-in-chief of the Journal of Computer-Aided Design & Computer Graphics (in Chinese).