Efficient Identification of Multi-Cycle False Path

Kai Yang, Kwang-Ting Cheng
University of California, Santa Barbara
{kyang, timcheng}@ece.ucsb.edu

Abstract

Due to false paths and multi-cycle paths in a circuit, using only topological delay to determine the clock period could be too conservative. In this paper, we address the timing analysis problem by considering both single-cycle and multi-cycle operations. We give a precise definition of multi-cycle false paths and provide the necessary conditions for multi-cycle sensitizable paths. We then propose an efficient algorithm to identify multi-cycle false paths. By considering both single-cycle and multi-cycle false paths, we could derive a shorter clock period than that determined by existing methods. Finally, we propose an algorithm to compute the valid clock period and demonstrate the improvement in clock frequency by taking multi-cycle false paths into account.

1 Introduction

The clock period of a circuit is determined by the delay of the longest path in the circuit. Static timing analysis, which computes path delay primarily based on topological delay, is an efficient approach for determining the valid clock period.

However, previous research shows that due to false paths [1–3] and multi-cycle paths [4–9] , it might be too conservative to use the topological delay for calculating the path delay. By taking into account both false paths and multi-cycle paths, we find that the timing constraints of the circuit could be relaxed.

The actual delay of a circuit is determined by the delay of its longest sensitizable paths. A false path is a path that cannot be sensitized by any input vector. The definition of false paths and algorithms for identifying all or a subset of them have been studied for years [1–3, 10–12]. In [10], Cheng and Chen proposed the notion of functional unsensitizable paths and demonstrated that those paths will not affect the circuit timing under *any* delay configuration.

A multi-cycle path in a sequential circuit is a combinational path which does not have to complete the propagation of the signals along the path within one clock cycle. A k-cycle path would have up to k clock cycles to propagate the transition from the source to the destination. The clock period can therefore be relaxed if the longest paths are multi-cycle paths. Figure 1 shows a circuit containing multi-cycle paths.

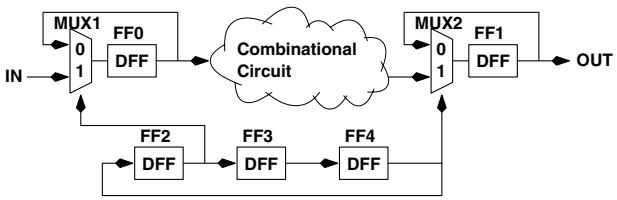

Figure 1: Example of Multi-Cycle Paths [5]

The initial values in flip-flops $\{FF_2, FF_3, FF_4\}$, which form an autonomous circular shift register, are $\{1, 0, 0\}$ respectively. Therefore, the sequence of states in these three flip-flops would be: $\{1, 0, 0\} \rightarrow \{0, 1, 0\} \rightarrow \{0, 0, 1\} \rightarrow \{1, 0, 0\}$. Multiplexer MUX1 selects primary input data IN when $\{FF_2, FF_3, FF_4\}$ are in state $\{1, 0, 0\}$. Then FF_0 latches the value IN and starts launching the new value toward the combinational circuit while $\{FF_2, FF_3, FF_4\}$ are switched to state $\{0, 1, 0\}$. Meanwhile, MUX2 selects the output of the combinational block when $\{FF_2, FF_3, FF_4\}$ are in state $\{0, 0, 1\}$. Then FF_1 latches the new value when $\{FF_2, FF_3, FF_4\}$ are switched to state $\{1, 0, 0\}$. Since it requires two cycles for $\{FF_2, FF_3, FF_4\}$ from $\{0, 1, 0\}$ to $\{1, 0, 0\}$. Therefore, all the paths from FF_0 to FF_1 are 2-cycle paths. That is, the circuit would allow 2 clock cycles to complete the propagation of transitions along these paths.

Multi-cycle paths have been studied for several years [4–9]. For microprocessor designs, the method proposed in [8] utilizes functional information such as instruction-set architecture (ISA), to extract multi-cycle paths. The proposed procedure, manually extracting the functional constraints, may not be scalable for larger and complex designs. The methods proposed in [5–7] are based on the stable-state analysis to identify multi-cycle flip-flop pairs. All paths between the multi-cycle flip-flop pairs are multi-cycle paths. Such stable-state analysis could be done by BDD [5], SAT [6], or ATPG [7] techniques.

However, considering signals' stable states only, which is the assumption made in all previous methods for multi-cycle path identification [5–7], may result in optimistic (and, thus, invalid) prediction of the clock period. In [7], the problem introduced by the presence of static hazards was described but no clear

0-7803-9451-8/06/$20.00

analysis was provided to address this problem.

In this paper, we first define the multi-cycle false paths and multi-cycle sensitizable paths. We then provide some necessary conditions for multi-cycle sensitizable paths. We use the functional sensitization criterion, introduced in [10], to check path sensitizability, so the static-hazards problem can be implicitly considered. Then, we address the problem of determining valid clock periods for circuits containing multi-cycle paths. We propose an efficient method for computing the valid clock period, which takes into account both single-cycle and multi-cycle false paths.

The rest of the paper is organized as follows. In Section 2, we review some representative approaches on this topic. In Section 3, we review the definitions to be used throughout this paper. In Section 4, we give the definition of multi-cycle paths. The necessary conditions for single-cycle and multi-cycle sensitizable paths are derived in Section 5. An efficient algorithm to identify the multi-cycle false path is introduced in Section 6. In Section 7, we address the problem of computing valid clock period. An iterative method to calculate the valid clock period is proposed. In Section 8, we show some experimental results, followed by the conclusions.

2 Background

A multi-cycle path in a sequential circuit is a combinational path which does not have to complete the propagation of the signal transition from the source to the destination of the path in single clock cycle. A k-cycle path is a path that is allowed to use k clock cycles to propagate the transition. By considering multi-cycle paths, the timing constraints could be relaxed. That is, the minimal clock period could be shorter than the delay of the multi-cycle path.

To analyze multi-cycle operations, the method proposed in [6, 7] is based on checking the stable states in flip-flop pairs. A flip-flop pair (FF_i, FF_j) is classified as a multi-cycle flip-flop pair, if there exist input vectors to satisfy Equation 1, where $FF_i(t)$ denotes the stable state in flip-flop i at cycle t:

$$FF_i(t) \neq FF_i(t+1) \Rightarrow FF_j(t+1) = FF_j(t+2) \quad (1)$$

All paths between flip-flop pairs (FF_i, FF_j) are then declared as multi-cycle paths. However, because the stable-state checking only checks the necessary conditions for multi-cycle paths, it might not result in correct classification of multi-cycle flip-flop pairs due to the presence of static-hazards [7]. The problem was pointed out in [7] but no solution was provided. Moreover, the stable-state checking could not determine the multiplicity, k, of the identified multi-cycle paths. None of the previous work has yet addressed the problem of determining the valid clock period for circuits with multi-cycle paths.

In the following section, we first give the notation and definition of path sensitization used throughout the paper.

3 Definition

A path $P^x = (G_0, f_0, G_1, f_1, ..., f_{m-1}, G_m)$ is a sequence of gates and signals associated with a transition $x \in \{rising, falling\}$ at the source of the path. Gate G_0 is either a primary input or the output of an FF, G_m is either a primary output or the input of an FF. Signal $f_i, 0 \leq i \leq m-1$ is an **on-input** of P^x which connects gate G_i to G_{i+1}. A signal is called a **side-input** of P^x associated with f_i if it is connected to G_i, but not originated from G_{i-1}. The delay of a path P^x is denoted as $d(P^x)$. A path set η contains all paths in a circuit.

A state of a circuit, denoted as S, indicates the values in all or a subset of flip-flops at certain cycle. The controlling (non-controlling) value of gate G is denoted by $cv(G)$ ($ncv(G)$).

Let $v =< v_1, v_2 >$ be an input vector pair and assume v_2 becomes stable at time $t = 0$. The logic value stabilized at a signal f or the output of a gate G is called its stable value under v_2. The time when the value at a signal f or the output of a gate G becomes stable is called its stable time under v_2.

The latest arrival time of signal f for value x, as $ar^x_{max}(f)$, is the delay of the longest path from any primary input or the output of an FF to signal f for a corresponding transition. Similarly, the earliest arrival time is denoted as $ar^x_{min}(f)$.

3.1 Path Sensitization

Signal f, which is connected to G, is considered to **dominate** G if the stable value and the stable time at G are determined by those at f. A path is considered to be sensitized, under a delay configuration, by a vector pair if each on-input of the path dominates its connected gate. Given a delay configuration, a path is a **true** (**sensitizable**) path if there exists at least one vector pair which sensitizes the path. Otherwise, it's a **false** path.

Under a delay configuration, a vector pair sensitizes a path *iff* each on-input of the path is either the earliest controlling value or the latest non-controlling input with all its side-inputs being non-controlling inputs. This criterion is called the **exact sensitization criterion** [2].

To efficiently check the sensitizability of target paths, a delay-independent method, using the **functional sensitization criterion**, is proposed in [10]. If there exists an input vector v such that all the side-inputs of f_i along P^x settle to non-controlling values on v when the on-input f_i has a non-controlling value, then P^x is functional sensitizable. Otherwise, P^x is functional unsensitizable.

Because functional sensitization, a delay-independent criterion, is only a necessary condition of exact sensitization, an identified functional sensitizable path might not be sensitizable under certain delay configurations. On the other hand, the identified functional unsensitizable paths must be false under any arbitrary delay configuration.

4 Multi-Cycle Path

In this section, we give a precise definition of multi-cycle path and introduce a model for illustration and analysis.

A k-cycle path P^x could complete the propagation of the signal transition from the source to the destination in k cycles. This implies that the clock period clk could be shorter than the delay of P^x. The transition, along P^x, travels through a segment seg_i in cycle i ,and then continues propagating through segment seg_{i+1} in the next cycle. A k-cycle path can therefore be divided into several non-overlapping segments seg_i where $P^x = \cup(seg_i)$ for $1 \leq i \leq k$. The delay $d(seg_j)$ of each segment seg_j, for $1 \leq j \leq k-1$, is equal to the clock period clk. The delay of the last segment seg_k, $d(seg_k)$ is equal to $modulo(d(P^x), clk)$. Each segment seg_i represents the segment through which the transition travels in cycle i. Since every segment needs to consist of integral number of stages (gates), $d(seg_j)$ might not perfectly match clk. In this case, we ignore the gates which are overlapped for sensitization checking.

We use the **timeframe expanded model** of the sequential circuit to illustrate how a transition propagates through a multi-cycle path. Segment seg_i in timeframe-i (TF-i) represents the transition propagation path in cycle i. Figure 2 shows an example of a 2-cycle path. The same path appears in both TF-1 and TF-2. The transition is launched at the source of the path in TF-1. The transition travels through segment seg_1 in TF-1 (bold line shown inside TF-1) and then continues to travel through seg_2 to reach the destination in TF-2 (bold line shown inside TF-2).

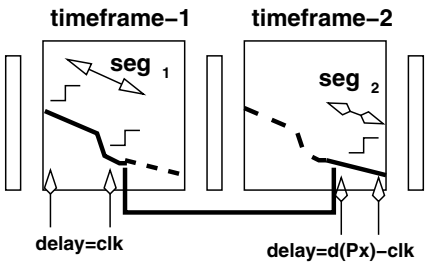

Figure 2: Example of 2-Cycle Path

5 Necessary Conditions for Path Sensitization

Due to the delay variation, the sensitizability of a path might be different from chip to chip. To identify paths which are false under any delay configuration, we derive necessary conditions , which are delay-configuration independent, for both single-cycle and multi-cycle path sensitization. Paths which cannot satisfy these necessary conditions are false paths.

5.1 Single-Cycle Paths

The following is a necessary condition for a single-cycle sensitizable path: A single-cycle sensitizable path P^x must satisfy the functional sensitization criterion [10]. Otherwise, P^x is false.

The set of single-cycle false paths, under a delay configuration M and a state S, is denoted as $\zeta_1^{M,S}$. The set of other paths which do not belong to $\zeta_1^{M,S}$ is denoted as $\omega_1^{M,S}$. That is, $\omega_1^{M,S} \cup \zeta_1^{M,S} = \eta$ and $\omega_1^{M,S} \cap \zeta_1^{M,S} = \phi$.

5.2 Multi-Cycle Paths

The following is a necessary condition for a multi-cycle sensitizable path: Under the timeframe expanded model, each segment seg_i of a multi-cycle sensitizable path P^x must satisfy the functional sensitization criterion in its corresponding timeframe. Otherwise, P^x is false.

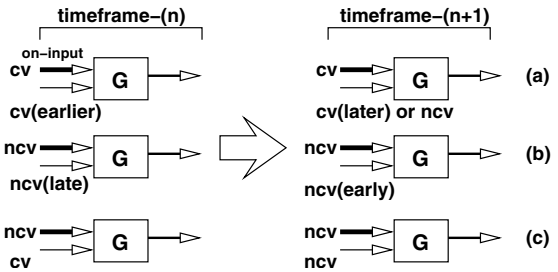

Figure 3: Side-input dominates gate G in TF-(n)

We prove the necessary condition for a multi-cycle sensitizable path by contradiction. We show that if the propagated transition is blocked in any segment (i.e. the functional sensitization criterion is violated), then the transition cannot continue propagating through the multi-cycle path (i.e. it's a multi-cycle false path). The left hand side of Figure 3 enumerates all possible conditions that a transition, propagating through multi-cycle path P^x and arriving at gate G in timeframe-(n), would be blocked by the side-input. The right hand side shows the values of on-input and side-input of gate G in timeframe-($n+1$). In the first case, as shown in Figure 3 (a), the on-input propagates a controlling value which arrives at gate G in timeframe-(n). If the side-input has a controlling value which arrives earlier than the on-input, then the side-input dominates gate G in timeframe-(n). In this case, regardless of the side-input value in timeframe-($n+1$), either non-controlling or controlling, the output of gate G will not change its value in timeframe-($n+1$) (i.e. no

transition occurs at gate G). Therefore any transition propagated through this gate must be dominated by the side-input. For cases shown in Figure 3 (b)(c), the on-input propagates a non-controlling value which arrives at gate G in timeframe-(n). If the side-input has a controlling value (case (c)), or a non-controlling value which arrives later than the on-input (case (b)), then the side-input dominates gate G in timeframe-(n). In this case, even if the side-input becomes a non-controlling value in timeframe-($n+1$), the output of gate G either will not change its value or the new value will be dominated by the side-input in timeframe-($n+1$). Therefore, we can conclude that, in order to propagate a transition along a multi-cycle path P^x, each segment needs to be sensitizable in its corresponding timeframe.

The set of k-cycle false paths, under a delay configuration M and an initial state S, is denoted as $\zeta_k^{M,S}$. The initial state S indicates the state at which the transition is launched from the source of P^x. The set of other paths which do not belong to $\zeta_k^{M,S}$ is denoted as $\omega_k^{M,S}$. That is, $\omega_k^{M,S} \cup \zeta_k^{M,S} = \eta$ and $\omega_k^{M,S} \cap \zeta_k^{M,S} = \phi$.

6 Identification of Multi-Cycle False Paths

In this section, we introduce a segment-based checking algorithm to identify multi-cycle false paths. The algorithm checks the necessary condition described in Section 5.2. Under the timeframe expanded model, the algorithm, summarized in Algorithm 1, checks the sensitizability of each segment of the multi-cycle path at each timeframe. The inputs of the algorithm are a path or a partial path P^x, the multiplicity k, and the clock period clk.

Algorithm 1 Segment-Based Checking(P^x, k, clk)

```
For each f, calculate ar^x_max(f), for x = {0,1}
Construct timeframe expanded model comb^k for P^x
for i = 1, i ≤ k, i = i + 1 do
  // check the sensitization of all side-inputs in seg_i
  for each side-input f connected to G in seg_i do
    if clk ≥ ar^{ncv(G)}_max(f) then
      if on-input of G propagates an ncv(G) then
        impose ncv(G) to f at timeframe-i of comb^k
      end if
    end if
  end for
end for
Perform logic implication. If conflict, P^x is false.
Otherwise P^x is true
```

We first calculate the latest arrival time for value $x = 0$ and $x = 1$ for each signal f, which is denoted as $ar^x_{max}(f)$. We then construct the timeframe expanded model $comb^k$ and divide P^x into non-overlapping segments. The delay of each segment seg_j for $1 \leq j \leq k-1$ is clk. The delay of the last segment seg_k is $modulo(d(P^x), clk)$.

We check the sensitizability of P^x segment by segment through $comb^k$. For each segment seg_i, the sensitization constraints are imposed, followed by implication, on the expanded circuit $comb^k$ in its corresponding timeframe i. Note that the side-inputs of seg_i may not reach its stable value in timeframe i because a side-input may also be part of a multi-cycle path. Therefore, instead of imposing all logic constraints required for functional sensitization of seg_i, we impose constraints only at side-inputs which can reach stable value in timeframe i. For example, if value $ncv(G)$ is needed at side-input f sensitizing segment seg_i in timeframe-i, and the latest arrival time of f for value $ncv(G)$,$ar^{ncv(G)}_{max}(f)$, is smaller than clk (which means signal f would become stable at $ncv(G)$ within this cycle) , then we impose value $ncv(G)$ at f in timeframe-i of $comb^k$. Otherwise, no side-input constraint is imposed. Figure 4 shows an example of the segment-based checking for a 2-cycle path. To sensitize the multi-cycle path P^x, signals a and b need to be at logic "1" in timeframe-1. Similarly, to sensitize the segment in timeframe-2, signals c and d need to be at logic "1".

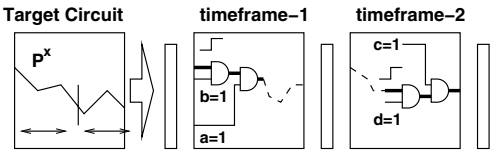

Figure 4: Example of Segment-Based Checking

After imposing the sensitization constraints, we perform logic implication on $comb^k$ to see whether there is any conflict. If a conflict occurs, P^x is false. Otherwise it classified as true. Please note that the identified false paths must be false. However, because we only check a necessary condition, the rest may or may not be true.

7 Valid Clock Period

Traditionally, the valid clock period is determined by the delay of the longest single-cycle sensitizable path. However, due to the presence of multi-cycle paths, the clock period might be relaxed. Equation (2) summarizes the constraints for the valid clock period in presence of multi-cycle operations.

$$clk \geq (\max(d(P^x))/m, \forall P^x \in \omega_m^{M,S}), 1 \leq m \leq k \quad (2)$$

Based on Equation (2), we propose a method to calculate the valid clock period for circuits with multi-cycle paths.

7.1 Calculation of Valid Clock Period

Using an exemplar circuit architecture, shown in Figure 5, as the driver, illustrate the proposed iterative method for determining the valid clock period. In Figure 5, flip-flops $\{FF_{c0}, FF_{c1}, ..., FF_{cn}\}$ forms an autonomous circular shift register. The transitions at the inputs of the combinational sub-circuit are launched when the state in $\{FF_{c0}, FF_{c1}, ..., FF_{cn}\}$ switches from $\{1, 0, 0, ..., 0\}$ to $\{0, 1, 0, ..., 0\}$.

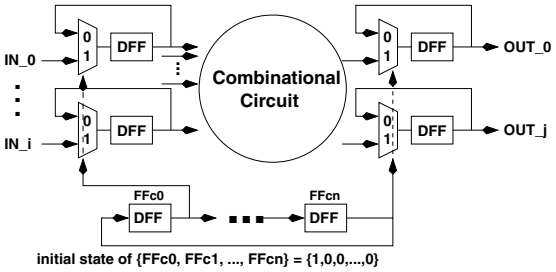

Figure 5: Exemplar Circuit Architecture

We make the following observations regarding the output-MUX in Figure 5, whose gate-level implementation is shown in Figure 6. The select-line (sel) of the output-MUX always has a controlling value of gate G_1 when $\{FF_{c0}, FF_{c1}, ..., FF_{cn}\}$ are not in state $\{0, 0, 0, ..., 1\}$. In addition, the latest arrival time of the select-line is always earlier than the earliest arrival time of In_1. Therefore, based on the exact sensitization criterion, transitions traveling through In_1 to gate G_1 will always be blocked by the select-line when $\{FF_{c0}, FF_{c1}, ..., FF_{cn}\}$ are not in state $\{0, 0, 0, ..., 1\}$.

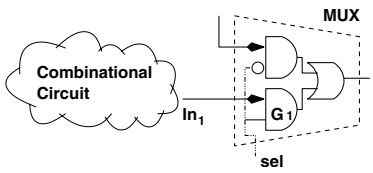

Figure 6: Output MUX

To calculate the valid clock period for a multi-cycle circuit similar to the one in Figure 5, we propose an iterative method which is shown in Figure 7. In each iteration, we identify the longest k-cycle true path and update the circuit maximum delay MAX. The valid clock period will then be determined by the final MAX.

In the preprocess step, we first identify all signals connected to the output-MUXs (such as signal In_1 in Figure 6) and group these signals as FS. The iterative procedure then identifies the longest single-cycle $(k = 1)$ true path, which starts from the output of an FF and a PI. To efficiently identify the longest true path, we utilize the Best-First-Search (BFS) method [11] to identify the longest path. We then check

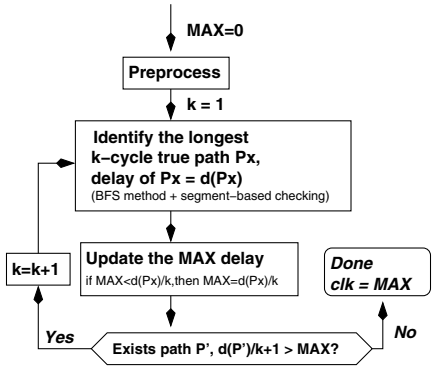

Figure 7: Calculation of Valid Clock Period

the sensitizability using the segment-based checking algorithm. The inputs to the checking procedure are the path/partial-path identified by the BFS , k, and the delay of the path/partial path. In addition to checking functional sensitizability, we also utilize FS (the set of signals that we group in the preprocessing phase) to help identify the false paths. Assume the propagated transition of a path P^x arrives at signal f in cycle-i, at which $\{FF_{c0}, FF_{c1}, ..., FF_{cn}\}$ are not in state $\{0, 0, 0, ..., 1\}$. If the f belongs to FS, then P^x is false. Once the longest single-cycle sensitizable path is identified, its delay is then assigned to MAX.

In the next iteration, we first identify paths P whose delay divided by $k + 1$ are longer than MAX. If no such path exists, the process stops and MAX is the minimal valid clock period. Otherwise, we check $k+1$-cycle sensitizability of each path in P. If a path in P is identified as a $k + 1$-cycle true path and its delay divided by $k + 1$ is longer than MAX, then we update MAX. After all paths in P are processed, we continue on to the next iteration.

8 Experimental Result

For the experiments, we constructed 2-cycle operation circuits by replacing the combinational sub-circuit in Figure 5 with ISCAS-85 circuits and constructed the autonomous circular shift register with three registers $\{FF_{c0}, FF_{c1}, FF_{c2}\}$. The initial values in flip-flops $\{FF_{c0}, FF_{c1}, FF_{c2}\}$ are $\{1, 0, 0\}$ respectively.

For the given circuits, we compare the resulting clock periods for three different methodologies: static timing analysis, single-cycle-false-path-aware timing analysis [12] ,and the proposed method which considers both single-cycle and multi-cycle false paths. All the methods are implemented by C++ and run on Linux workstations. Table 1 summarizes the experimental results.

The first column shows the name of the circuit used as the combinational sub-circuit in Figure 5. The second column shows the delay of the longest

path of the circuit which is calculated by static timing analysis, denoted as clk_{sta}. The third column shows the valid clock period derived by the longest single-cycle sensitizable path, denoted as clk_{sc} The fourth column shows the resulting clock clk_{mc} calculated by the proposed method, and the CPU time for computing the valid clock period is listed in the fifth column. As indicated, the clock period derived by the proposed method is shorter than those derived by traditional methods.

Table 1: Valid Clock Period

Replace-ckt	$clk_{sta}(ns)$	clk_{sc}	clk_{mc}	$CPU(s)$
c17	13.4	13.4	6.7	0.0
c1355	135.2	135.2	67.6	0.11
c1908	184.8	184.8	92.4	0.07
c2670	125.7	125.7	62.8	0.11
c3540	226.1	224.8	112.4	0.48
c432	110.5	107.4	53.7	0.6
c449	135.4	135.4	67.7	0.55
c5315	216.6	202.0	101.0	1.92
c7552	188.6	181.7	90.8	1.77
c880	109.6	109.6	54.8	0.04

The reason for the conservative result of clk_{sc} is that the traditional methods do not take the circuit states into account. It's assumed that all primary inputs and outputs of FFs could have any value. So some paths classified as false by the proposed method are classified true in traditional methods.

Previous work [7] addressed the problem of sampling error in presence of static hazards. Such a problem will not occur if the clock period is determined by the proposed method. If a static-hazard is propagated, then it must go through at least one true path. In the proposed method, the clock period is derived from the longest true path of the circuit, so the static-hazard will disappear, and the signals should become stable before a FF latches its data.

9 Conclusion

In this paper, we first define the multi-cycle false paths and multi-cycle sensitizable paths. We then provide the necessary conditions for multi-cycle sensitizable paths. We use the functional sensitization criterion, introduced in [10], to check the path sensitizability, so the static-hazards problem, ignored in previous approaches, can be implicitly considered. Then, we provide a thorough analysis for the problem of determining valid clock period for circuits containing multi-cycle paths. We have proposed an algorithm to compute the valid clock period and demonstrate the improvement to clock frequency by considering multi-cycle false paths.

References

[1] P. McGeer and R. Brayton, *Efficient Algorithms for computing the Longest Viable Path in a Combinational Network*. Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1989.

[2] H.-C. Chen, D. H.-C. Du, and L.-R. Liu, "Critical path selection for performance optimization", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, pp. 185–195, Feb. 1993.

[3] S. Davadas, K. Keutzer, S. Malik, and A. Wang, "Certified timing verification and the transition delay of a logic circuit", *IEEE Trans. VLSI Systems*, vol. 2, no. 3, pp. 333–342, Sept. 1994.

[4] A. P. Gupta and D. P. Siewiorek, *Automated Multi-Cycle Symbolic Timing Verification of Microprocessor-based Designs*. Proc. IEEE/ACM Design Automation Conf. (DAC), 1994.

[5] K. Nakamura, K. Takagi, S. Kimura, and K. Watanabe, *Waiting False Path Analysis of Sequential Logic Circuits for Performance Optimization*. Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), Nov. 1997.

[6] K. Nakamura, S. Maruoka, S. Kimura, and K. Watanabe, *Multi-Cycle Path Detection based on Propositional Satisfiability with CNF Simplification using Adaptive Variable Insertion*. IEICE Trans. on Fundamentals, Dec. 2000.

[7] H. Higuchi, *An Implication-based Method to Detect Multi-Cycle Paths in Large Sequential Circuits*. Proc. IEEE/ACM Design Automation Conf. (DAC), June 2002.

[8] W.-C. Lai, A. Krstic, and K.-T. Cheng, "Functionally testable path delay faults on a microprocessor", *IEEE Design & Test of Computers*, vol. 15, 2000.

[9] P. Ashar, S. Dey, and S. Malik, *Exploiting multi-cycle false paths in the performance optimization of sequential circuits*. Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1992.

[10] K.-T. Cheng and H.-C. Chen, "Classification and identification of nonrobust untestable path delay faults", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 845–853, Aug. 1996.

[11] W. Qiu and Walker D.M.H., *An efficient algorithm for finding the k longest testable paths through each gate in a combinational circuit*. Proc. Int. Test Conf. (ITC), 2003.

[12] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, *False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation*. Proc. IEEE/ACM Design Automation Conf. (DAC), 2002.