Verifying Clock Schedules in the Presence of Cross Talk

Soha Hassoun* Christopher Cromer ° Eduardo Calvillo-Gámez*
Tufts University* Infineon Technologies Corp.°
{soha,calvillo}@eecs.tufts.edu Christopher.Cromer@Infineon.com

Abstract

This paper addresses verifying the timing of circuits containing level-sensitive latches in the presence of cross talk. We show that three consecutive periodic occurrences of the aggressor's input switching window must be compared with the victim's input switching window. We propose a new phase shift operator to allow aligning the aggressor's three relevant switching windows with the victim's input signals. We solve the problem iteratively in polynomial time, and show an upper bound on the number of iterations equal to the number of capacitors in the circuit. Our experiments demonstrate that eliminating false coupling results in finding a smaller clock period at which a circuit will run.

1. Introduction

Shrinking process geometries have posed new challenges in static timing analysis. Most notably, capacitances due to lateral (same layer) and overlapping (different layers) couple to each other and affect the circuit speed. Furthermore, the change in delay is dependent on the exact switching scenario of the considered victim net and the aggressor nets – the neighboring nets that capacitively couple to the victim. If the victim is stable while the aggressors change, then the victim net might cause a functional error. If, however, the victim is switching in the same direction as the aggressor(s), then we have *assistive coupling*, and the victim switches sooner than anticipated. With *opposing coupling*, the victim net switches later due to an opposing transition on the aggressor(s). Static timing analysis here consists of comparing the switching windows of the inputs to the victim and to the aggressor to find possible overlaps.

Big strides in modeling cross talk and in timing analysis in the presence of cross talk have been made. Driver modeling using reduced order modeling and worst case alignment of the aggressors relative to the victim determine the worst case victim delay [2, 4, 11]. Some iterative techniques have been proposed [1, 8], and it was demonstrated that they converge after few iterations. A theoretical foundation of the correctness of iterative techniques and the possibility of multiple solutions has been provided [13].

This paper addresses cross talk analysis for circuits with level-sensitive latches. Level-clocked circuits are certainly dominant in high-performance designs because they can operate at faster clock rates than edge-triggered circuits [3]. This is because, unlike edge-triggered registers, latches allow borrowing time across their boundaries. Researchers have efficiently solved the problem of verifying a clock schedule [12]. Naively assuming worst case cross talk while running these algorithms yields pessimistic schedules.

The difficulty of the problem is due to the periodic nature of signals in a sequential circuit. Coupling can occur because of the proximity of a victim's switching input to any *periodic* occurrence of the aggressor's input switching window. In addition, arrival times are not computed in absolute time as in combinational circuits. In a level-sensitive circuit, however, all signal arrival times are computed relative to local time zones depending on the phases of nearby latches. Translating values from the aggressor's to the victim's time zones is thus needed to determine overlap in switching windows at the inputs of the victim and aggressor.

This paper begins by reviewing the gate delay and clock model. Next, timing equations and coupling conditions are derived. Then, we present a polynomial algorithm to verify the timing of a level-clocked circuit when given a clock schedule. We conclude with experimental results.

2 Preliminaries

2.1 Clock Schedule Model

Our clock schedule model is based on the SMO formulation [7]. A n-phase clock schedule is an ordered collection of n periodic signals, $(\phi_1, ..., \phi_n)$, having a common period π. Because phases are periodic, a *local time zone* of width π is associated with each phase. Each phase ϕ_i is characterized by two parameters e_i and w_i. Parameter e_i

represents the absolute time when ϕ_i begins (relative to an arbitrary global time reference). Parameter w_i is the length of time that ϕ_i is active (latch is open). To translate one measurement a from the local time zone of ϕ_i into the *next* local time zone of ϕ_j, we subtract from a a phase shift operator $E_{i,j}$, defined as:

$$E_{i,j} = \begin{cases} e_j - e_i & \text{if } i < j \\ \pi + e_j - e_i & \text{otherwise.} \end{cases}$$

This clocking scheme is demonstrated in Figure 1. We assume that the design intention and thus the clock schedule specify that a signal departing from a latch k must be captured by the next latching edge, occuring after the latching edge of k, of the following latch l. The earliest arrival time at the output of a latch k clocked by ϕ_i is $\pi - w_i$, and it must arrive at the input of the following latch l clocked by ϕ_j on or before latch l's closing edge: $\pi + E_{i,j}$ time units after the beginning of ϕ_i. We ignore setup and hold times to simplify our presentation.

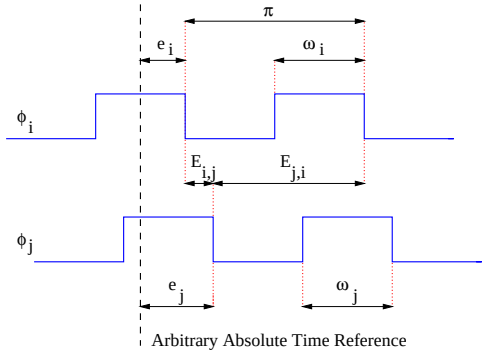

Figure 1. An example clock schedule that illustrates the SMO clocking model.

2.2 The Delay Model

While several choices are possible for modeling the added delay due to coupling, we choose to use the dynamically bounded delay model [5], an abstract model that allows a gate's delay to be assigned one of many values depending on related operating conditions. This model captures most delay variations within the fixed range $[\delta, \Delta]$, while explicitly modeling all other variations. With a narrower fixed range, more explicit variations must be represented. With a wider range, only a few variations must be represented. If all variations are captured with the $[\delta, \Delta]$ range, then our model is essentially the commonly used fixed, or *min-max*, delay model. Delays associated with cross coupling are modeled as follows: Assume that the output of a node, a, capacitively couples to the output a node v. With opposing coupling, v's maximum delay is increased by a $\Delta_{v,a}$. With assistive coupling, v's minimum delay is decreased by a value $\delta_{v,a}$. A predicate indicates when this increase or decrease must hold.

2.3 Circuit Model

A circuit is modeled as a directed graph $G = (V, E, C)$. Each vertex in V represents either a primary input, primary output, a combinational gate, or a latch. The set of all combinational gates is referred to as V_C, and the set of all latches is referred to as V_L. P_v refers to the set of predecessors to node $v \in V$.

Each edge in E represents the connectivity between two vertices. C represents the set of capacitors in the circuit. A set C_v is the set of aggressor nodes connected via a capacitor to node v. Each node v has a dynamically bounded delay model consisting of a fixed delay range $[\delta_v, \Delta_v]$. In addition, for each coupling capacitance attached to v and an aggressor node a, four delay values: $\Delta_{v,a}, \Delta_{a,v}, \delta_{v,a}$, and $\delta_{a,v}$, and a predicate indicates when the conditional delays should be considered.

We designate the latest (earliest) arrival time at a node v as A_v (a_v). The latest (earliest) departure time at a node is denoted by D_v (d_v). The time reference of D_v and d_v is based on associating each node v with a *phase*, $p(v)$, which is derived by analyzing the phases of the latches in the combinational fan in and fan out of node v.

3 Timing Equations

The earliest and latest arriving signals at the inputs of the victim and aggressor must be analyzed to determine if the switching windows overlap. The latest arrival time at at a combinational node v:

$$A_v = \begin{cases} max_{\forall_{k \in P_v}}(D_k - E_{p(k),p(v)}) & \text{if } p(k) \neq p(v) \\ max_{\forall_{k \in P_v}} D_k & \text{if } p(k) = p(v) \end{cases} \quad (1)$$

For a latch v with input k:

$$A_v = max(D_k - E_{p(k),p(v)}, \pi - w_{p(v)}) \quad (2)$$

Here the latest arrival time at the latch depends on the relative arrival time of the signals at its input, D_k, and when the latch allows the data through, $\pi - w_{p(v)}$. If the input signal k arrives before the latch is open, then it must wait till the latch opens before k is passed through. The departure time D_k is adjusted by $E_{p(k),p(v)}$ to transfer the departure time of k to v's local time zone.

The departure time from a node v, without capacitive coupling on its output, can be specified as follows:

$$D_v = A_v + \Delta_v \quad (3)$$

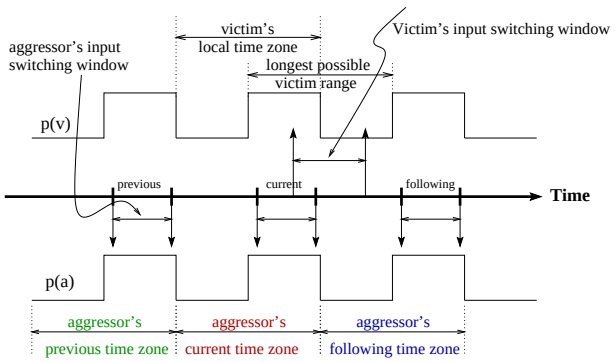

Figure 2. When the aggressor's and victim's time zones are aligned, we must check for overlap between the switching windows of the inputs to the victim and aggressor.

To compute D_v, the departure time at v, we augment the latest arriving input to v by an amount Δ_v, the maximum propagation delay through v.

For a node v with capacitive coupling on its output through one or more aggressor in C_v, the maximum departure time is:

$$D_v = A_v + \Delta_v + \sum_{\forall a \in C_v} \gamma_{v,a} \Delta_{v,a} \tag{4}$$

This constraint ensures that the propagation delay of v is augmented by an amount $\Delta_{v,a}$ when a node v (the victim) experiences capacitive coupling through an aggressor a. Worst case opposing coupling between v and a is assumed because we are not considering the functional/logical behavior of the circuit. Variable $\gamma_{v,a}$ is binary indicating if the conditions for capacitive coupling hold. A description of conditions that cause coupling is provided in Section 4.

We similarly specify constraints for minimum arrival and departure times. For a combinational node v:

$$a_v = \begin{cases} min_{\forall_{k \in P_v}} (d_k - E_{p(k),p(v)}) & \text{if } p(k) \neq p(v) \\ min_{\forall_{k \in P_v}} d_k & \text{if } p(k) = p(v) \end{cases} \tag{5}$$

For a latch v:

$$a_v = max(d_k - E_{p(k),p(v)}, \pi - w_{p(v)}) \tag{6}$$

The earliest departure time for a node v can be specified as follows assuming worst case assistive coupling between a victim node v and an aggressor a.

$$d_v = a_v + \delta_v - \sum_{\forall a \in C_v} \gamma_{v,a} \delta_{v,a} \tag{7}$$

4. Coupling Conditions

Due to the periodicity of signals in a sequential circuit, coupling can occur due to the overlap, or close proximity by an amount of τ, of the switching window at the input of the victim and any *periodic* switching window at the aggressor's input.

Consider the situation depicted in Figure 2, where the aggressor and the victim have the same phase, $p(v) = p(a)$, resulting in aligned time zones. Considering the maximum possible victim range, determining the overlap in the switching windows is straight forward, and it is essentially that same as for combinational circuits, namely:

$$max(a_v, a_a) \leq min(A_v, A_a) + \tau$$

However, when $p(a) \neq p(v)$, it is possible for the switching window associated with the victim's input to overlap with either none, one, or two of the three possible switching windows: the *previous*, *current*, and/or *following* switching windows of the aggressor's input. The previous occurrence can be computed by subtracting π from the range, (i.e. $[A_a - \pi, a_a - \pi]$), while computing the *following* occurrence requires adding π.

When $p(v) \neq p(a)$, the arrival times at the input of the aggressor must be translated to the victim's local time zone to perform meaningful comparisons. We designate the aggressor's *current* time zone is the *closest* in time from victim's local time zone. The *previous* aggressor's time zone is the one preceding the *current* aggressor's time zone. The *following* aggressor's time zone is the one succeeding the *current* aggressor's time zone.

To determine the *closest* aggressor time zone, we compare the positions of $p(a)$ and the $p(v)$. Recall from Section 2.1 that the phases are ordered periodic signals, and that each is associated with a parameters e_i, the time when phase i begins relative to an absolute reference point.

If the victim's time zone leads or lags the aggressor's local time zone by or less than $\pi/2$, (i.e. $-\pi/2 \leq e_{p(v)} - e_{p(a)} \leq +\pi/2$), then the latter time zone is designated as the *current* time zone. If the victim's time zone leads (occurs before) the aggressor's local time zone by more than $\pi/2$, (i.e. $e_{p(v)} - e_{p(a)} < -\pi/2$), then the latter is designated as a *following* time zone. Similarly, if the victim's time zone lags (occurs after) the aggressor's local time zone by more than $\pi/2$, (i.e. $e_{p(v)} - e_{p(a)} > \pi/2$), then the latter time zone is designated as *previous*. An example situation is depicted in Figure 3.

To translate a value local to the aggressor's time zone to the victim's time zone and to have that value appear as a *current* occurrence, we define a new phase shift operator, $E'_{i,j}$, as follows:

$$E'_{i,j} = \begin{cases} e_j - e_i + \pi & \text{if } e_j - e_i < -\pi/2 \\ e_j - e_i & \text{if } -\pi/2 \leq e_j - e_i \leq +\pi/2 \\ e_j - e_i - \pi & \text{if } e_j - e_i > +\pi/2 \end{cases}$$

This operator differs from the SMO phase shift operator, $E_{i,j}$, which always translates a value in ϕ_i's local time zone

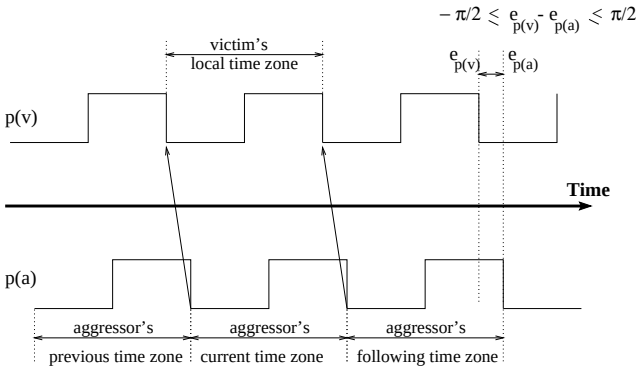

Figure 3. The *current* aggressor's time zone is always defined as the *closest* in time from the victim's local time zone.

forward in time into the *next* local time zone of ϕ_j – where *next* is defined to be the one with the following latching edge.

Based on our analysis and our new operator, we can now define coupling to occur in the following cases:

- coupling with the current occurrence:
  $max(a_v, a_a - E'_{a,v}) \leq min(A_v, A_a - E'_{a,v}) + \tau$
- coupling with the following occurrence:
  $A_v + \tau \geq a_a - E'_{a,v} + \pi$ *and* $A_v > a_a - E_{a,v}$
- coupling with the previous occurrence:
  $a_v \leq A_a + \tau - E'_{a,v} - \pi$ *and* $a_v \leq A_a - E_{a,v}$

If and only if one of the above coupling conditions holds, then the binary $\gamma_{v,a}$ is set to one.

5. Algorithm

Our algorithm for verifying that a circuit runs correctly for a given clock schedule is iterative. Initially, all coupling is assumed not to hold: all $\gamma_{v,a}$ variables are set to zero. During each iteration, the steps below are performed. This algorithm is run until no new γ variables are assigned.

1. Compute the latch-to-latch, PI to latch, and latch-to-PO minimum and maximum delays as outlined in [9]. The run time is dominated by $O(|L| \times (|V| + |E|))$.
2. Using the delays computed in step 1, run the Szymanski/Shenoy [12] algorithm to compute the arrival times and identify the switching windows at the inputs of victims and aggressors. The run time of the algorithm is $O(|L|^3)$. A post-processing step computes arrival and departure times to compare the proper switching windows. It's run time is linear in the number of circuit nodes and edges.
3. Compare the switching windows as outlined in the previous section, and set the appropriate binary γ variables. The run time is linear in the number of nodes, assuming a small number of aggressors is associated with each victim.

The Szymanski/Shenoy algorithm utilizes latch-to-latch delays, thus the first step is necessary to ensure the efficiency of the latter algorithm. During each iteration, the latch-to-latch delays are recomputed because new γ variables are assigned.

Our presented algorithm is guaranteed to converge. Once a new γ is assigned, the victim's window is simply stretched (the A_v becomes larger, and the a_v becomes smaller). Such a change in the victim's window can only cause other windows to further stretch. This argument of continually shrinking or expanding switching windows was used to prove convergence for timing analysis for combinational circuits [1, 8, 13]. The algorithm is guaranteed to converge in $|C|$ iterations because, in the worst case, one γ is assigned true through each iteration. Once $|C|$ iterations are completed no switching windows can possibly stretch.

6. Experimental Results

Our experiments evaluate the effectiveness of our algorithm in verifying clock schedules in the presence of cross talk. Due to the lack of relevant public benchmarks, we created our own. Our benchmarks, summarized in table Table 1, are based on a subset of the edge-triggered MCNC FSM circuits that were first converted to circuits with level-sensitive latches. We used SIS to perform the mapping [10]. We then converted registers to back-to-back phi1/phi2 latches and used sskew [6] to determine an equal, two-phase retiming and an initial clock schedule. The combinational nodes in the circuit were initialized with a maximum random delay within 2.5 and 0.5; the minimum delay was then initialized with a random value that is at most 0.5 less than the maximum delay. We then added random capacitors equal in number to 10% of the total circuit nodes. Each capacitor was assigned a random delay between 0 and 1.0. Next, we ran sskew to determine a worst case symmetric schedule. Table 1 lists the *maximum period* that assumes worst case capacitive coupling, and the normalized *m*inimum period, which assumes zero coupling.

To find the minimum clock period while accurately analyzing coupling, we ran our program to test several equal, two-phase schedules in a logarithmic fashion, each time eliminating half the range between the last verified minimum and maximum clock schedules. The first clock period tested was the minimal period that can be obtained when ignoring coupling. After that, the clock period used was equal to the average between the maximum and minimum

Circuit	Primary Inputs	Latches	Combinational gates	Maximum Period	Minimum Period Normalized	Final Period Normalized	# Iterations	Total Run Time (sec)
bbsse	7	8	87	13.93	0.903	0.903	1	0.21
cse	7	8	153	14.08	0.939	0.939	1	0.34
dk16	2	11	189	14.34	0.890	0.904	6	2.22
ex2	2	10	100	14.12	0.892	0.898	5	0.96
ex6	5	17	89	13.54	0.947	0.954	5	0.97
sand	11	86	436	14.49	0.938	0.938	1	3.28
kirkman	12	8	174	14.98	0.875	0.875	1	0.42
train4	2	4	13	7.204	0.905	1.000	4	0.07

Table 1. Results Table. The first three columns present the number of primary inputs, latches, and combinational gates. The Maximum Period column presents the maximum clock period assuming worst case capacitive coupling; the Minimum Period column reports the minimum possible period ignoring all capacitive coupling. The Final Period is obtained using our iterative technique. The number of iterations indicates how many schedules were attempted.

periods. This process was repeated until the difference in periods between two successive was less than or equal to 0.1.

From our results in Table 1, we see that only one circuit operated at the maximum clock period (train4). The circuit has a combinatorial delay from a primary input to a primary output that sets the clock period. For a few others, the circuit ran at a smaller clock period than the maximum one. Often, the circuit was able to run at the indicated minimum clock period. The number of iterations required until the analysis converged was small compared to the number of the capacitors in the circuit. The run time (for all iterations) for each case is reported running on a Sun Enterprise-250.

7. Conclusions

This is the first paper that addresses cross talk analysis for circuits with level-sensitive latches. The main contributions of this paper are (a) showing the coupling conditions necessary to detect coupling, (b) deriving a new phase shift operator to conveniently translate the aggressor's periodic occurrences to the victim's local time zone, and (c) a polynomial algorithm to solve timing verification for level-sensitive circuits in the presence of cross talk. Our experiments demonstrate that eliminating false coupling effects can reduce the clock period at which a circuit will run.

References

[1] R. Arunachalam, K. Rajagopal, and L. Pileggi. "TACO: Timing Analysis With COupling". In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, pages 576–80, 2000.

[2] F. Dartu and L. Pileggi. "Calculating Worse-Case Gate Delays Due to Dominant Capacitance Coupling". In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, pages 46–51, 1997.

[3] C. Ebeling and B. Lockyear. "On the Performance of Level-Clocked Circuits". In *Advanced Research in VLSI*, pages 242–356, 1995.

[4] P. Gross, R. Arunachalam, K. Rajagopal, and L. Pileggi. "Determination of Worst-Case Aggressor Alignment for Delay Calculation". In *Proc. of the IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 212–9, 1998.

[5] S. Hassoun. "Critical Path Analysis Using a Dynamically Bounded Delay Model". In *Proc. of ACM-IEEE Design Automation Conf.*, 2000.

[6] B. Lockyear. *Algorithms for Retiming Level-Clocked Circuits and their use in Increasing Circuit Robustness*. PhD thesis, University of Washington, Seattle, Washington, 1994.

[7] K. Sakallah, T. Mudge, and O. Olukotun. Analysis and design of latch-controlled synchronous circuits. In *Proc. 27th ACM-IEEE Design Automation Conf.*, 1990.

[8] S. Sapatnekar. A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing. *IEEE Transactions on Computer-Aided Design*, 19(5):550–559, May 2000.

[9] S. Sapatnekar and R. Deokar. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE Transactions on Computer-Aided Design*, 15(10):1237–48, October 1996.

[10] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. "SIS: A System for Sequential Circuit Synthesis". Technical Report UCB/ERL M92/41, University of California, Dept. of Electrical Engineering and Computer Science, May 1992.

[11] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo. "Driver Modeling and Alignment for Worst-Case Delay Noise". In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2001.

[12] T. Szymanski and N. Shenoy. "Verifying Clock Schedules". In *Proc. of the IEEE International Conference on Computer-Aided Design (ICCAD)*, pages 124–131, November 1992.

[13] H. Zhou, N. Shenoy, and W. Nicholls. "Timing Analysis with Crosstalk as Fixpoints on Complete Lattice". In *Proc. of the ACM/IEEE Design Automation Conference (DAC)*, 2001.