Timing Verification with Crosstalk for Transparently Latched Circuits

Hai Zhou
Electrical and Computer Engineering
Northwestern University
Evanston, IL 60208

ABSTRACT

Delay variation due to crosstalk has made timing analysis a hard problem. In sequential circuits with transparent latches, crosstalk makes the timing verification (also known as clock schedule verification) even harder. In this paper, we point out a false negative problem in current timing verification techniques and propose a new approach based on switching windows. In this approach, coupling delay calculations are combined naturally with latch timing iterations. A novel algorithm is given for timing verification with crosstalk in transparently latched circuits and primitive experiments show promising results.

Categories & Subject Descriptors

B.7.2 Design Aids: Verification

General Terms

Algorithms, Design, Verification

Keywords

Timing, Clock, Verification, Coupling, Delay

1. INTRODUCTION

With increasing clock frequencies and shrinking process geometries in deep sub-micron technology, both capacitive and inductive crosstalk become big concerns in designs. Besides introducing noises on quiet wires, crosstalk may greatly change the wire delays. Even worse, the change of a delay by crosstalk is data-dependent, that is, it can increase or decrease the delay based on the switching directions of the two coupled signals. These data-dependent delay variations present a new challenge on timing analysis.

There are many recent works that deal with the timing analysis with crosstalk. Since whether two coupled signals will change each other's delay is dependent on whether they may switch at (approximately) the same time, timing analysis and crosstalk effect calculation are mutually dependent on each other. To solve this chicken-and-egg problem, iterative approaches are proposed to bring them into a consistent state [10, 1, 3, 13]. Zhou et al. [13] established the mathematical foundation of timing analysis with crosstalk as a fixpoint on a complete lattice and pointed out different ways to find a fixpoint (among possibly many fixpoints). These works are mainly focused on timing analysis of a combinational circuit. However, they can be easily carried on to sequential circuits using only edge-triggered registers. Because signals cannot pass transparently through edge-triggered registers, timing analysis needs to be conducted only on the combinational component of such a circuit. The output times of the registers are used as the input times of the combinational circuit, and the set-up and hold conditions of the registers are checked against the combinational outputs.

However, many sequential circuits have not only edge-triggered registers but also level-sensitive latches. Especially in high-performance designs, level-clocked circuits are dominant since level-sensitive latches take less area and operate at higher frequencies than edge-triggered registers [5]. The complexity introduced by the latches is that signals can now pass transparently through the latches and this makes time borrowing across latch boundaries possible. Therefore, timing analysis can no longer be carried only on the combinational part since the output time of a latch is now dependent on its input time. Szymanski and Shenoy [12, 11] provided a now standard technique to solve the problem without considering crosstalk. They separated the computation of early arrival times from that of late arrival times and formulated each of them as a fixpoint computation. In each problem, an iteration is started from a lower bound initial solution to generate a monotonic increasing sequence.

In the presence of crosstalk, Hassoun et al. [7] directly combined the Szymanski and Shenoy algorithm with Hassoun's dynamically bounded delay model [6] to handle timing verification. Because of crosstalk, the early and late arrival times of a signal can no longer be separated. However, the Szymanski and Shenoy algorithm uses constant latch-to-latch path delays which are no longer valid in the presence of crosstalk. This introduced a false negative problem in the timing verification. These problems will be explained in more details in Section 3.

To deal with the above problems, we propose a totally new approach to solve the timing verification for transparently latched in the presence of crosstalk. As a basis for our solution, the timing verification problem without crosstalk is first revisited. The traditional approach (such as Szymanski and Shenoy) can be viewed as a steady state computation and thus may introduce a false negative problem on hold conditions. When crosstalk is presented, the false negative problem may propagate to set-up conditions. To overcome these problems, we give a new formulation of accumulating switching windows which can be viewed as computing all

states reachable from the initial state. Since crosstalk effects can be computed by increasing the switching windows, in the presence of crosstalk, iterations for crosstalk and iterations for latches work in the same fashion and thus can be combined in any order. Based on the chaotic iteration scheme [2], we design an efficient algorithm for the verification which exploits the flexibility in the order to speed up the computation. Primary experimental results show that our approach is correct and promising.

2. PRELIMINARIES

2.1 Models of clock and transparent latches

Clock is the most important mechanism used in a synchronous circuit. A clock is a periodical signal used to regulate other signals in the circuit. With the help of a clock, the design of other signals is relaxed to tolerate glitches or false transitions if they pass the safety checking of the clock. Multiple phases of a clock may be used in a circuit. A *clock scheme* for a circuit is a set of periodical signals $\phi_1, \ldots, \phi_n$ with a common period c. A three phase clocking scheme is shown in Figure 1. Selecting a period of length c as the *global time reference*, we can denote each phase ϕ_i by its starting and ending times (s_i, e_i) with respect to the reference. Note that it is possible to have $s_i > e_i$ based on the selection of global time reference. We generally order the phases such that $e_i < e_j$ if $i < j$. Also note that w_i is used to represent the width of phase i.

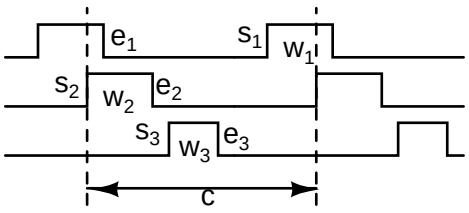

Figure 1: Three phase clocking with period c.

Memory elements are used in a circuit to store its state. In a synchronous circuit, clock signals are usually applied at memory elements to do the safety checking and to filter out unintended transitions. For this purpose, besides the data input and output, a synchronous memory element has a control input. Only under a certain condition of the control input does the memory element respond to the data input. Memory elements can be categorized into two groups according to how they respond to the control input: *flip-flops* store the data when the control switches; *latches* let the output have the input value when the control level is high. Figure 2 shows the symbol used for a memory element and the signal responses in a latch and a flip-flop. As we can see, when a clock signal is applied to the control input of a memory element, transitions not synchronous with the clock are filtered out. A flip-flop filters out all transitions while a latch let the transitions within the high level window of the control pass transparently to the output. Because of this, timing verification is easy in a circuit with only flip-flops but is very hard when latches are used. We only focus on the latter problem in the sequel.

2.2 Szymanski and Shenoy algorithm

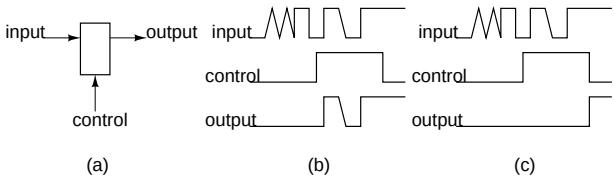

Figure 2: (a) A memory element; (b) Signal response in a latch; (c) Signal response in a flip-flop.

Szymanski and Shenoy's method for timing verification (also known as clock schedule verification) [12] is based on a simple topological delay model of the combinational component. Given the pin-to-pin delay of each gate, the maximal and minimal delays from the output of one latch to the input of another latch can be computed by traversing the topology of the gate connections. Let Δ_{ij} and δ_{ij} to represent the maximal and minimal combinational delays from latch i to latch j, respectively. Also let A_i and a_i represent the later and early signal arrival times on the input of latch i, and D_i and d_i the later and early signal departure times on the output of latch i, respectively. Based on the SMO formulation [9], they have

$$\begin{aligned} A_i &= \max_{j \to i}(D_j + \Delta_{ji} - E_{p_j p_i}) \\ a_i &= \min_{j \to i}(d_j + \delta_{ji} - E_{p_j p_i}) \\ D_i &= \max(A_i, c - w_{p_i}) \\ d_i &= \max(a_i, c - w_{p_i}) \end{aligned}$$

Where p_i is the clock phase controlling latch i and E_{ij} is defined as

$$E_{ij} = \begin{cases} e_j - e_i & \text{if } e_j > e_i \\ c + e_j - e_i & \text{otherwise} \end{cases}$$

We must also note that used here are local times referring to local periods that end with the phase falling edges.

Szymanski and Shenoy verified the clock schedule by first solving the above equations and then checking whether the solution satisfies the set-up and hold conditions

$$\begin{aligned} A_i &\leq c - S_{p_i} \\ a_i &\geq H_{p_i} \end{aligned}$$

The equations are solved by an iterative approach starting from a lower bound, that is

$$\begin{aligned} a_i^0 &= -\infty \\ A_i^0 &= -\infty \\ d_i^m &= \max(a_i^m, c - w_{p_i}) \\ D_i^m &= \max(A_i^m, c - w_{p_i}) \\ a_i^m &= \min_{j \to i}(d_j^{m-1} + \Delta_{ji} - E_{p_j p_i}) \\ A_i^m &= \max_{j \to i}(D_j^{m-1} + \delta_{ji} - E_{p_j p_i}) \end{aligned}$$

3. FALSE NEGATIVE IN PREVIOUS APPROACHES

Hassoun et al. [7] used the Szymanski and Shenoy algorithm directly as a subroutine in their timing verification in the

presence of crosstalk. It works as follows. At the beginning, no crosstalk is assumed to take effect and the Szymanski and Shenoy algorithm is used to find a solution. Then the solution is used to check for switching window overlap and modify the delays based on crosstalk effects. These two processes are repeated until there is no change on delays.

Without crosstalk, there is already a false negative problem in the Szymanski and Shenoy algorithm. As the early arrival and departure times increase monotonically to their steady-state values, they might very well violate hold conditions at various latches even if the steady-state solution will not. This has been noted by Szymanski and Shenoy [12] and can be easily fixed by checking the hold conditions at the first iteration.

However, in the presence of crosstalk, this minor false negative problem on hold conditions may underestimate switching window overlap, and thus introduce false negative on set-up conditions. This can be exemplified through a simple circuit in Figure 3(a). There are four latches $1, 2, 3, 4$ clocked by two non-overlap phases. The clock period is $3d$ where d is the gate delay of an inverter. We also assume that when the two coupled signals switch in the opposite directions at the same time, their delays will be increased by $0.5d$. In Hassoun et al.'s approach, no coupling is assumed to take effect at the beginning of the iteration. The Szymanski and Shenoy algorithm is used under the assumption that the delay from latch 1 to latch 2 is $2d$ and that from latch 3 to latch 4 is d. The converged solution of the latch arrival times by the Szymanski and Shenoy algorithm is shown in Figure 3(b). Notice that since there is only one path to each latch, the early and late arrival times of the latch coincident with each other. Since the departure windows at latch 1 and latch 3 do not overlap, the switching windows of the two coupled signals do not overlap with each other. This means that the solution in Figure 3(b) is a converged solution by Hassoun et al and the clock schedule is valid.

However, this verification is false negative. Consider an initial state with value 0 at latches 1, 2, 4, and value 1 at latch 3. The violation trace is shown in Figure 3(c). When the clock starts ticking, latches 1 and 3 will have their output switch at the rising edge of ϕ_1 (shown by the two shorter arrows in Figure 3(c)). This will make the coupled signals switch in opposite directions at the same time, thus will increase the delays by $0.5d$. The arrival times at latches 2 and 4 are thus delayed by $0.5d$, which are shown by the shorter arrows. In the second clock period, the arrival times at latches 1 and 3 are both delayed by $0.5d$, thus still make the coupled signals switch at the same time. The arrival times for all the latches are shown by the longer arrows. As we can see, the set-up condition at latch 2 has already been violated.

This false negative problem on set-up conditions is more serious than that on hold conditions and cannot be easily fixed. One of its causes is a theoretical pitfall in Hassoun et al.'s approach: a constant latch-to-latch delay model is used within the Szymanski and Shenoy algorithm; however, it is not valid in the presence of crosstalk.

For example, consider the combinational part from latches

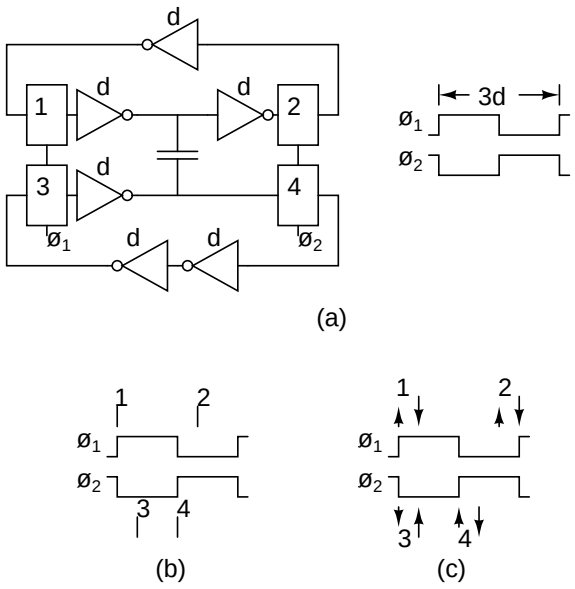

Figure 3: False negative in Hassoun et al.'s approach. (a) The circuit and clock schedule; (b) Solution by Hassoun et al.; (c) A violation trace.

1 and 3 to latches 2 and 4 in Figure 3, redrawn in Figure 4. The constant path delay model needs to assign a delay value from a to x and a delay value from b to y. If we use $2d$ for the first path and d for the second path, we underestimate the delays when a and b switch at the same time; if we use $2.5d$ for the first path and $1.5d$ for the second path, we overestimate when a and b switch at different time. Since the initial state and the steady state in the Szymanski and Shenoy algorithm may have different timing behaviors, using constant path delay model causes problems.

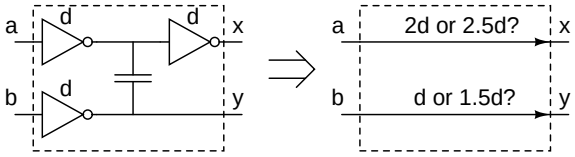

Figure 4: Constant path delay model no longer works in the presence of crosstalk.

4. A NEW FORMULATION BASED ON SWITCHING WINDOWS

In this section, we revisit the timing verification for transparently latched without crosstalk. Even though the minor false negative on hold conditions can be easily fixed, we will investigate the deep theoretical reason for its cause and propose a sound formulation based on switching windows. This new formulation may look equivalent to the simple fix when there is no crosstalk, but its benefit will show up in the presence of crosstalk.

Szymanski and Shenoy [12] directly followed Sakallah et al. [9] to solve the timing verification problem by first solving the SMO equations and then checking the set-up and hold

conditions. However, none of them proved the correctness of this approach. Szymanski and Shenoy discovered that there may be multiple solutions to the SMO equations and, in that case, not all of the solutions satisfy the set-up and hold conditions. They also noted the false negative problem on hold conditions. These problems should have raised enough suspicions on the SMO formulation, but the tradition was still followed.

Zhou et al. [13] established a firm foundation for timing analysis in the presence of crosstalk, which is based on the general understanding of timing behavior as an approximation of the circuit semantics. A key element in this understanding is that the information we seek is not on a specific run but *on all possible runs.* That is, the information should be a subset of instances instead of just one instance. Since what needs to be checked in the timing verification is the stability of each latch input signal from the set-up time to the hold time, the subset of all possible switching times of the signal needs to be computed.

Based on the operation model of a latch, any signal transition in the input during the high level of the clock will appear on the output and *any stable transition before the clock high level will generate a transition at the rising edge of the clock.* Therefore, if the switching times on the latch input fall in $[a_i, A_i]$, then the switching times on the latch output are in $[d_i, D_i]$, where

$$\begin{aligned} d_i &= \max(a_i, c - w_{p_i}) \\ D_i &= \max(A_i, c - w_{p_i}) \end{aligned}$$

which has been characterized by the SMO formulation. However, these equations only compute the sets of switching times in the next state from those in the current state. That is, they are the counterparts of the transition functions in a finite state machine. Iterating through them only gives the timing information of the *steady* states[1]. Since the timing verification needs to check the timing safety of all possible states, we need to accumulate all possible switching times. Let $\alpha_i = [a_i, A_i]$ denote the current subset of possible switching times on the input of latch i and δ_i the current subset of possible switching times on the output of latch i. Then, the subset of possible switching times on the output should be increased as

$$\delta_i' = \delta_i \cup [\max(a_i, c - w_{p_i}), \max(A_i, c - w_{p_i})]. \quad (1)$$

Intuitively, this formulation is similar to the union of all next states to find the reachable states in model checking [8].

In (1), when δ_i is an interval and $\delta_i \cap [\max(a_i, c - w_{p_i}), \max(A_i, c - wp_i)] \neq \emptyset$, the result δ_i' is still an interval. Otherwise, the representation of δ_i' will become complicated. However, under the constant path delay model, we can always expand a complicated set δ_i' to the minimal interval covering it without introducing false positive problems on set-up and hold conditions.

Similar to Zhou et al. [13], if we use the set inclusion relation ($\subseteq$) as a partial order on the solution space, the iteration function (1) is monotonic[2]. Specifically, the subset of switching times on any signal is non-decreasing during the iterations and thus will either converge to a fixpoint or violate the set-up and hold conditions. Contrary to Sakallah et al. [9] and confirmed with Szymanski and Shenoy's observation [12], the selection of initial solution is very important here. Usually, each latch has a reset input which can be used to reset the latch into an initial state. By the worst case principle, we assume that the next state will have a different bit value on each latch from the initial one. This means that initially there is always a transition on each latch output at the rising edge of its clock phase. Started with this initial solution, larger and larger subsets are computed till they converge or violate the conditions. Checking a fixpoint against the setup and hold conditions will make sure that the conditions are not violated *at any point during the circuit operation.*

5. TIMING VERIFICATION WITH CROSSTALK

Our approach to timing verification with corsstalk in transparently latched circuits is based on the new timing window based formulation presented in Section 4. In this approach, the verification is no longer carried out on the early and later arrival times separately. Instead, switching windows are accumulated during the iterations. Because of this, the constant path delay model is no longer used. Furthermore, since the switching windows are non-decreasing in the new timing verification, *the window update by crosstalk can be combined naturally with the window update by latch feedbacks.* Specifically, our algorithm, called XVeriClock, works as follows. In the beginning, all the latch outputs have their switching windows of length 0 at their clock rising edges, all the primary inputs have their given windows, and all other signals have empty windows. Then switching windows are iteratively updated over the whole circuit through three kinds of window propagations: the output window of a gate or a wire has a constant delay from its input windows; the switching window of a wire is changed by the switching windows of its coupled wires; the switching window of a latch output is updated through (1). It can be verified that all these window update functions are monotonic, that is, if the input window does not get reduced, then the output window cannot become smaller. For a more formal treatment of monotonicity, see [13]. The advantage of (1) is the guarantee that the latch output windows are monotonically non-decreasing during the iterations. This, combined with the monotonicity of window update functions, assures that all windows are non-decreasing during the iterations. Therefore, the iterations will either converged to a fixpoint or reach a point where setup or hold condition is violated. The pseudo-code of the algorithm XVeriClock is given in Figure 5.

Actually the above algorithm XVeriClock is just an algorithm scheme. It does not give the order to de-queue the windows in Q. Even for a given window w in Q, the sequence to update the influenced windows may not be "crosstalk, fanout, and then latch" as shown in the pseudo-code. Since all the updates will increase the switching windows, based

[1]Since iterating through transition functions cannot guarantee convergence on states, the convergence of the Szymanski and Shenoy algorithm should be considered a surprise. Further study shows that the constant path delay model plays a critical role in Szymanski and Shenoy's convergence proof.

[2]On the contrary, the SMO formulation is not monotonic on this partial order set.

```
Algorithm XVeriClock

set all switching windows to empty;
window queue Q = ∅;
initialize primary input windows
  and put them in Q;
set latch output windows to points at
  clock rising edges and put them in Q;
while (Q ≠ ∅ and not stop) {
   de-queue a window w from Q;
   if (w gets overlap with coupling windows)
      update coupling windows;
   if (w has fanouts)
      update fanout windows;
   if (w is a latch input)
      update the latch output window;
   add all changed windows to Q;
}
```

Figure 5: The algorithm XVeriClock

on the scheme of chaotic iteration [4, 13], the iterations will reach the same result–either diverge or converge to the same fixpoint–no matter what order of updates is applied. This can be stated as the following theorem.

THEOREM 1. *If the algorithm XVeriClock converge to a fixpoint in one update order, it will converge to the same fixpoint in any update order.*

The flexibility given in the update order can be further exploited to speed up the computation. The fanout relations through gates and latches form directed edges between signals. Coupled signals form two directed edges between them in opposite directions. Similar to Zhou et al. [13], if strongly connected components are identified and processed one by one in their topological order, some of the useless updates will be skipped. However, since many latches form feedbacks in a sequential circuit, we may have larger strongly connected components here. Within each strongly connected component, there are many different ways to arrange the update order but there is no obvious winner [2]. Generally a strongly connected component is viewed as a DAG (directed acyclic graph) plus a set of feedback edges. Even though finding the minimal set of feedback edges is NP-hard, a small set of feedback edges is usually desired. The method to propagate through the feedback edges as soon as possible is called the *recursive approach*; the method to propagate through the feedback edges as later as possible is called the *iterative approach* [2].

However, the edges in our system are not homogeneous. The edges representing couplings have very different property from those representing gate or latch fanout relations. Because of this, Szymanski and Shenoy's upper bound on the number of iterations to convergence [12] is no longer valid when there is crosstalk. Fortunately the following result can be proved.

THEOREM 2. *If the algorithm XVeriClock converges, it must converge within n iterations after the last window update from crosstalk, where n is the number of latches.*

PROOF. Here, each iteration means updating each signal once. Since there is no delay changed by crosstalk in the last n iterations, the path delay from a latch to another is constant. Using results in Szymanski and Shenoy [12], the iterations will diverge if they do not converge within n iterations. □

Because of the above theorem, we prefer to update coupling windows as soon as possible, in the hope to find convergence or divergence as early as possible. After each coupling window update, the number of iterations will be re-counted and used as one of the stop criteria in the algorithm.

6. EXPERIMENTAL RESULTS

Different from Hassoun et al. [7], our approach is built directly upon accumulated switching windows, and thus does not need to use the Szymanski and Shenoy algorithm and the constant path delay model with it. This helps to solve the false negative problems in Szymanski and Shenoy (minor one) and in Hassoun et al. (major one). Furthermore, the approach also makes it possible to combine coupling updates with latch updates. Our primary experiments are used to test how large the switching windows could be under the accumulated model. Combining the updates from couplings and latches can greatly reduce the running time and is now under testing.

Thanks to Hassoun, we got the test cases and the source code used in their experiments [7]. In our primary experiments, we change the iterations to accumulate switching windows (i.e., using (1) instead of the SMO equations) but keep the update order the same. We use the same parameters, e.g. the same random delays of the combinational gates and random capacitors equal in number to 10% of the total nodes. For each of the test cases, a max and a min periods are checked by our approach and Hassoun et al.'s approach. When an approach diverges, we also give the violating window. The result is shown in Table 1.

7. CONCLUSIONS

Delay variation due to crosstalk has become an important issue that must be taken into consideration in all timing analyses and verifications of DSM designs. Combined with the time borrowing across latch boundaries, it makes the timing verification with crosstalk in transparently latched circuits an important and difficult problem. A false negative problem in the hold condition is revealed by our study of the SMO formulation and the traditional techniques without crosstalk. Directly using the traditional techniques with crosstalk will amplify the false negative problem and even get it into the set-up condition.

In this paper, we proposed a new approach based on accumulated switching windows to solve the timing verification problem. In this approach, switching windows are accumulated through the whole operation of the circuit thus no false negative problem will happen. Furthermore, coupling window updates can be combined naturally with window updates through gates and latches. This flexibility is used

Table 1: Experimental Results

circuit				max period		min period	
name	PIs	latches	gates	Hassoun	ours	Hassoun	ours
bbsse	7	8	87	pass	pass	pass	pass
cse	7	8	153	pass	pass	pass	pass
dk16	2	11	189	pass	pass	[8.252,19.274]	[6.38,19.274]
ex2	2	10	100	pass	pass	[9.321,18.984]	[6.3,18.984]
ex6	5	17	89	pass	pass	[12.11,19.336]	[12.11,19.336]
kirkman	12	8	174	pass	pass	pass	pass
train4	2	4	13	pass	pass	[5.331,10.464]	[5.331,10.464]

to speed up the computation in a novel algorithm and the primary results are promising.

Acknowledgments

The author is grateful to S. Hassoun for providing the code of their algorithm. This research was supported by a faculty start-up fund from Northwestern University.

8. REFERENCES

[1] R. Arunachalam, K. Rajagopal, and L. T. Pilleggi. Taco: Timing analysis with coupling. In *Proc. of the Design Automation Conf.*, pages 266–269, Los Angeles, CA, June 2000.

[2] Francois Bourdoncle. Efficient chaotic iteration strategies with widening. In *International Conf. on Formal Methods in Programming and Their Applications, LNCS 735*, number 735 in LNCS, pages 128–141, 1993.

[3] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Switching window computation for static timing analysis in presence of crosstalk noise. In *Proc. Intl. Conf. on Computer-Aided Design*, San Jose, CA, November 2000.

[4] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, CA, January 1977.

[5] C. Ebeling and B. Lockyear. On the performance of level-clocked circuits. In *Advanced Research in VLSI*, pages 242–356, 1995.

[6] S. Hassoun. Critical path analysis uing a dynamically bounded delay model. In *Proc. of the Design Automation Conf.*, pages 260–265, Los Angeles, CA, June 2000.

[7] Soha Hassoun, Christopher Cromer, and Eduardo Calvillo-Gamez. Verifying clock schedules in the presence of cross talk. In *Proc. DATE: Design Automation and Test in Europe*, 2002.

[8] Kenneth L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, 1993.

[9] Karem A. Sakallah, Trevor N. Mudge, and Oyekunle A. Olukotun. *checkT_c* and *mint$_c$*: Timing verification and optimal clocking of synchronous digital circuits. In *Proc. Intl. Conf. on Computer-Aided Design*, pages 552–555, November 1990.

[10] S. 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*, 2000.

[11] Narendra V. Shenoy. *Timing Issues in Sequential Circuits.* PhD thesis, UC Berkeley, 1993.

[12] T. G. Szymanski and N. Shenoy. Verifying clock schedules. In *Proc. Intl. Conf. on Computer-Aided Design*, 1992.

[13] H. Zhou, N. Shenoy, and W. Nicholls. Timing analysis with crosstalk as fixpoints on a complete lattice. In *Proc. of the Design Automation Conf.*, pages 714–719, 2001.