Tradeoff Between Latch and Flop for Min-Period Sequential Circuit Designs With Crosstalk

Chuan Lin and Hai Zhou, *Senior Member, IEEE*

***Abstract*—Latches are extensively used in high-performance sequential circuit designs to achieve high frequencies because of their good performance and time-borrowing feature. However, the amount of timing uncertainty due to crosstalk accumulated through latches could be larger than the benefit gained by time borrowing. In this paper, we show that the tradeoff between a latch and a flop can be leveraged in a sequential circuit design with crosstalk, so that the clock period is minimized by selecting a configuration of mixed latches and flops. A circular time representation is proposed to make coupling detection easier and more efficient. Experiments on our heuristic algorithm for finding an optimal configuration of mixed latches and flops showed promising results.**

***Index Terms*—Algorithms, circuit optimization, crosstalk, design automation, timing, timing circuits, very large scale integration.**

I. Introduction

In A sequential circuit, clock signals are usually applied at memory elements to lock correct state values and to filter out unintended transitions. Generally speaking, memory elements can be categorized into two groups according to how they respond to clock signals: 1) edge-triggered *flops* store the data when the clock switches and 2) level-sensitive *latches* let the output have the input value during its active duration.

Latches dominate high-performance designs as they have smaller delays and occupy less area than flops. More importantly, since signals can pass through a latch anytime during its active duration, time borrowing between two consecutive latches is possible. As a result, circuits designed with latches can operate at higher frequencies than their edge-triggered counterparts [1].

For correct operation, the data are required to be stable before the latching edge by an amount of time called the *setup time*—This is known as the *setup condition*. The data have to remain stable after the latching edge for an amount of time called the *hold time*—This is known as the *hold condition*. A *clocking violation* refers to a violation of the setup condition or the hold condition.

Clock schedule verification (also known as timing verification) for edge-triggered circuits is straightforward. Since signals cannot pass transparently through flops, the output times of the flops are used as the input times of the combinational components, and the setup and hold conditions of the flops are checked against the combinational outputs. However, clock schedule verification involving latches is much harder. Since signals can pass through latches transparently during their active durations, the setup and hold conditions on paths between any two latches need to be considered. These conditions become even more complex in the presence of a multiphase clock schedule.

On the other hand, with increasing clock frequencies and shrinking process geometries in deep-submicrometer technology, both capacitive and inductive couplings (also known as crosstalks) become big concerns in designs. For present-day processes, the coupling capacitances can be as high as the sum of the area capacitances and the fringing capacitances. Trends indicate that the role of crosstalk will be even more dominant in the future as feature sizes shrink [2]. Besides introducing noises on quiet wires, crosstalk may greatly change the wire delays and, hence, affect the timing analysis. If an aggressor and a victim switch simultaneously in the same direction, the victim will speed up, which is called *assistive coupling*. Likewise, if an aggressor and a victim switch simultaneously in opposite directions, the victim will slow down, which is called *opposing coupling*. Assuming that coupling capacitances and inductances dominate all other capacitances and inductances of wires, failure to take crosstalk effects into timing analysis may produce wrong results. Our work in this paper equally applies to crosstalk induced by capacitive and inductive couplings. However, to simplify the presentation, we only talk about capacitive couplings in the rest of this paper.

Since latches allow signals to pass transparently, crosstalk effects on switching windows may accumulate on a long path and cause a clocking violation in the end. These phenomena will be explained in more detail in Section II. It is interesting to observe that the switching window at the output of a flop is just a single point or a switching window with width zero (for the ease of the presentation, we assume no clock skew uncertainty). A natural thought is to select a set of latches such that when they are replaced with flops, the crosstalk effects contributed by their original output switching windows are greatly reduced, and the previous clocking violations are removed.

Our contribution in this paper is twofold. First, we propose a circular time representation for coupling detection. We show that detecting coupling under the circular time representation is easier than state-of-the-art approaches used in [3] and [4]. Furthermore, since complicated phase translations are avoided, clock schedule verification under the circular time representation is more efficient. Second, we formulate the tradeoff

Manuscript received August 6, 2005; revised December 28, 2005, April 28, 2006, and July 21, 2006. This work was supported by the National Science Foundation under Contract CCR-0238484. This paper was recommended by Associate Editor S. Nowick.

C. Lin is with Magma Design Automation Inc., San Jose, CA 95110 USA (e-mail: clin@magma-da.com).

H. Zhou is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA.

Digital Object Identifier 10.1109/TCAD.2006.888273

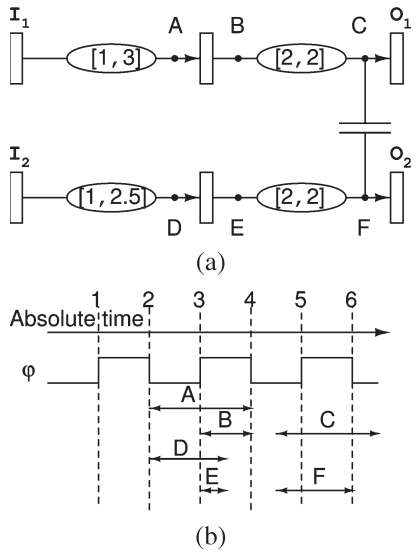


Fig. 1. Crosstalk effects on system performance.

between a latch and a flop in sequential circuits with crosstalk as a problem of seeking a configuration of mixed latches and flops to minimize the clock period. We present an effective and efficient heuristic algorithm to solve this problem. Experiments showed promising results.

The rest of this paper is organized as follows: Section II presents the motivation and problem formulation. Section III describes the models we will use in this paper. Section IV recaps the previous work. The circular time representation is proposed in Section V. Our algorithm is presented in Section VI, followed by experimental results in Section VII. Conclusions are given in Section VIII.

II. Motivation and Problem Formulation

We now use an example in Fig. 1(a) to show that the amount of timing uncertainty due to crosstalk accumulated through latches could be larger than the benefit gained by time borrowing. In this example, we have six latches driven by a clock ϕ with period 2 shown in Fig. 1(b), which has a duty cycle ratio of 0.5; i.e., the active duration is half the period. The ellipses between the latches represent the combinational blocks with their minimum and maximum delays. Opposing coupling increases the delay by 0.5, whereas assistive coupling decreases the delay by 0.5. Suppose that the setup and hold times are all zero and the inputs are available at I_1 and I_2 at time 1. Then, the switching windows of A and D are [2, 4] and [2, 3.5], respectively, as illustrated in Fig. 1(b). According to the transparent nature of latches, the switching windows of B and E are [3, 4] and [3, 3.5], respectively. If the coupling capacitor between C and F does not exist, then the switching windows of C and F will be [5, 6] and [5, 5.5], respectively. Since no clocking violation occurs, 2 is a feasible clock period. However, due to the presence of the coupling capacitor and the fact that the switching windows of B and E overlap, opposing and assistive couplings are assumed in the blocks they feed. As a result, the switching windows of C and F become [4.5, 6.5] and [4.5, 6], respectively. We detect a setup violation at latch O_1. Therefore, this circuit has to work at a period no smaller than 2.2 (assuming duty cycle scales proportionally with period).

However, if we replace the latch between A and B by a flop that operates at the falling edge of ϕ, then the switching window of B is shrunken to [4, 4]. Since the new window does not overlap with that of E, coupling is not triggered. Therefore, the switching windows of C and F become [6, 6] and [5, 5.5], respectively. The previous setup violation at O_1 is now removed. The new circuit can still work under period 2.

This example shows that the tradeoff between a latch and a flop can be leveraged to improve sequential circuit designs with crosstalk. We formulate it as a problem of finding a configuration of mixed latches and flops to minimize the clock period.

Problem 1 (Optimal Latch–Flop Configuration): Given a sequential circuit and its clock schedule, find a configuration of mixed latches and flops for the memory elements such that the circuit satisfies both setup and hold conditions with crosstalk under the minimum clock period.

III. Models and Notations

A. Circuit Model

A directed graph $G = (V, E)$ is used to represent a sequential circuit, where $V = V_G \cup V_L$ is the union of two sets of vertices, namely: 1) the gates V_G and 2) the memory elements V_L, and $E = E_I \cup E_C$ is the union of two sets of edges, namely: a) the interconnects E_I and b) the coupling capacitances E_C. Interconnect delays are available since we are considering circuits after placement and routing. We assume that all primary inputs and primary outputs are latched.

We designate the latest (earliest) arrival time at a vertex v as $A_v(a_v)$. The latest (earliest) departure time from a vertex is denoted by $D_v(d_v)$. $[a_v, A_v]$ and $[d_v, D_v]$ are called the input and output switching windows of v, respectively.

If v is a memory element, $p(v)$ denotes its given phase. Depending on the particular configuration, v could be a latch with phase $p(v)$ or a flop operating at the rising or falling edge of $p(v)$. For simplicity, we only talk about latches and the flops that operate at the falling edges. In the remainder of this paper, we simply use flops to refer to those that operate at the falling edges.

B. Delay Model

We choose to use the dynamically bounded delay model [5] to capture the delay variations due to crosstalk. More specifically, each vertex v has a combinational delay range $[\delta_v, \Delta_v]$. For each coupling capacitance between v and an aggressor u, if the switching window at v's input overlaps with that at u's input within a specified amount of time $\tau_{u,v}$, then opposing and assistive couplings are assumed at u and v. As a result, δ_v is decreased by $\delta_{u,v}$, and Δ_v is increased by $\Delta_{u,v}$. In this model, multiple aggressors are treated independently; i.e., their effects on victim are additive.

Although more accurate models are possible, such as [6]–[8], where superposition is used to calculate the total crosstalk effects without using coupling factors, the chosen model is a generalization of discrete coupling models, such as ones that assume a $0\times$, $1\times$, or $2\times$ effective coupling capacitance, e.g., [9].

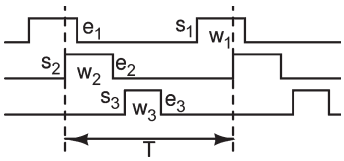


Fig. 2. Three-phase clock schedule with period T.

C. Clock Model

A *clock schedule* for a circuit is a set of periodic signals $\phi_1, \ldots, \phi_n$ with width w_i of the phase ϕ_i and a common period T. A three-phase clock schedule is shown in Fig. 2. Selecting a period of length T as the *global time reference*, we can order the phases ϕ_i by its starting times 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 the reference. We generally order the phases such that $e_i < e_j$ if $i < j$ and choose e_n as the global time reference. In particular, if $e_i = (i/n)T$, the clock schedule is *symmetric* [10]. The setup and hold times are denoted as X_i and H_i, respectively. We assume that both s_i and e_i scale proportionally with T, but X_i and H_i remain the same. A clock schedule is *valid* if and only if the circuit has no clocking violation under it. The common period of a valid clock schedule is called a *feasible* period. For simplicity, we assume zero clock skews; i.e., clock signals arrive at all the memory elements at the same time.

IV. Previous Work

Several techniques have been proposed to evaluate crosstalk effects on combinational delays. Some are based on iterative techniques [8], [9]; some are based on the propagation of events [11]; others are based on more complex mathematical formulations [5]. Consideration of the functional correlation of the victim and the aggressors allows further accuracy in analysis [12]–[14]. The worst case victim delay can be obtained by driver modeling using reduced-order modeling and worst case alignment of the aggressors relative to the victim [15]–[17]. What these techniques have in common is that crosstalk effects are expressed by expanding the switching window at the victim to accommodate more possible switchings.

The problem of clock schedule verification without considering crosstalk has been elegantly solved by Szymanski and Shenoy [18], [19]. Without crosstalk, we only need to consider switching windows at memory elements. The input switching window of a memory element i can be computed by propagating the output switching windows of i's fan-ins through shortest and longest path delays to i. Timing conditions for latches are formulated as [19]

$$A_i = \max_{j \to i} \left(D_j + C_{j,i} - E_{p(j)p(i)} \right) \tag{1}$$

$$a_i = \min_{j \to i} \left(d_j + c_{j,i} - E_{p(j)p(i)} \right) \tag{2}$$

$$D_i = \max \left(A_i, T - w_{p(i)} \right) \tag{3}$$

$$d_i = s_{p(i)} = T - w_{p(i)} \tag{4}$$

$$A_i \leq T - X_i \tag{5}$$

$$a_i \geq H_i \tag{6}$$

where $C_{j,i}$ and $c_{j,i}$ represent the maximum and minimum combinational delays from latch j to latch i, respectively, and $E_{p(j)p(i)}$ is a phase-shift operator defined as [20]

$$E_{p(j)p(i)} = \begin{cases} e_{p(i)} - e_{p(j)}, & \text{if } p(j) < p(i) \\ T + e_{p(i)} - e_{p(j)}, & \text{otherwise} \end{cases}.$$

Note that used here are local times referring to local time zones that end with the phase falling edges. Timing conditions involving flops are the same except for (3) and (4), where the output window of a flop is a single point $[T, T]$. We say that j is the *latest critical predecessor* of i if $A_i = D_j + C_{j,i} - E_{p(j)p(i)}$. If $a_i = d_j + c_{j,i} - E_{p(j)p(i)}$, then j is called the *earliest critical predecessor* of i.

We choose to characterize d_i by (4) instead of the more aggressive formulation with $d_i = \max(a_i, T - w_{p(i)})$ in [20]. This is because [21] showed that there are common situations, such as a latch driven by a qualified clock signal, in which the aggressive formulation is incorrect, and a similar problem arises in circuits that permit the clock to be stopped between adjacent latches to save power.

In the presence of crosstalk, however, switching windows at both memory elements and gates need to be considered because crosstalk may change the shortest and longest path delays between them.

Hassoun *et al.* [3] proposed to assign each gate v a phase, also denoted as $p(v)$, which can be derived by analyzing the phases of the memory elements in the combinational fan-in and fan-out of the gate. For example, $p(v)$ could be the phase of the clock driving the nearest memory element in v's combinational fan-in. Based on the phases at gates, the switching windows at gates can be computed in a similar fashion as those at memory elements, expressed as

$$a_v = \begin{cases} \min_{(u,v) \in E_I} \left(d_u - E_{p(u)p(v)} \right), & \text{if } p(u) \neq p(v) \\ \min_{(u,v) \in E_I} d_u, & \text{if } p(u) = p(v) \end{cases} \tag{7}$$

$$A_v = \begin{cases} \max_{(u,v) \in E_I} \left(D_u - E_{p(u)p(v)} \right), & \text{if } p(u) \neq p(v) \\ \max_{(u,v) \in E_I} D_u, & \text{if } p(u) = p(v) \end{cases} \tag{8}$$

$$d_v = a_v + \delta_v \tag{9}$$

$$D_v = A_v + \Delta_v. \tag{10}$$

Their approach of verifying clock schedules with crosstalk 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 compute the switching windows at gates by (7)–(10). The results are used to check for switching window overlaps and to modify the delays based on crosstalk effects. These three processes are repeated until there is no change on delays. Zhou [4] proposed an improved algorithm that essentially combined the three processes together and computed the accumulated switching windows during the iterations.

However, detecting overlap in both Hassoun *et al.*'s and Zhou's approaches was not easy. In [3], Hassoun *et al.* showed that when two vertices couple, the victim's input window can overlap with either one, two, or three of the three possible switching windows at the aggressor's input: the previous, the

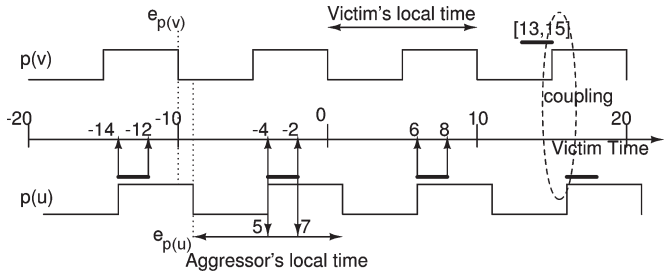


Fig. 3. Example where coupling is missed by $E_{p(u)p(v)}$.

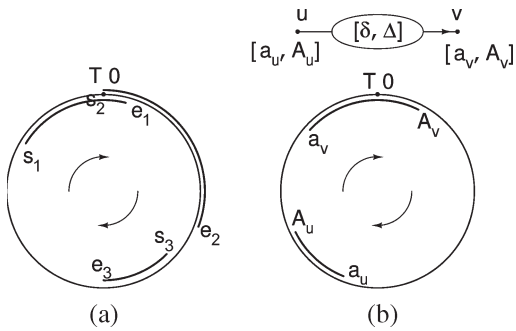


Fig. 4. (a) Circular time representation of the clock schedule in Fig. 2. (b) Switching window propagation through a combinational vertex under the circular time representation.

current, and the following windows. The input window of the aggressor must be translated to the victim's local time zone to perform a meaningful comparison. They showed that the previously defined phase-shift operator $E_{p(j)p(i)}$ cannot be used for such a translation since it may lead to coupling detection misses. Consider the example in Fig. 3 (taken from [3]), where $T = 10$, $e_{p(u)} - e_{p(v)} = 1$, $E_{p(u)p(v)} = 9$, $\tau_{u,v} = 1.01$, and 50% duty cycle. If $E_{p(u)p(v)}$ is used to translate the aggressor's ranges, i.e., the previous $[5-T, 7-T]$, the current [5, 7], and the following $[5+T, 7+T]$, then the ranges become $[-14, -12]$, $[-4, -2]$, and [6, 8], respectively. If the victim occurrence is [13, 15], then comparing the translated aggressor ranges against the victim's will not indicate a coupling problem. However, coupling should have been detected because the fourth occurrence [16, 18] is within $\tau_{u,v}$ of the victim occurrence.

As a remedy, they proposed a new phase-shift operator $E'_{p(j)p(i)}$, defined as

$$E'_{p(j)p(i)} = \begin{cases} e_{p(i)} - e_{p(j)} + T, & \text{if } e_{p(i)} - e_{p(j)} < -\frac{T}{2} \\ e_{p(i)} - e_{p(j)}, & \text{if } -\frac{T}{2} \le e_{p(i)} - e_{p(j)} \le +\frac{T}{2} \\ e_{p(i)} - e_{p(j)} - T, & \text{if } e_{p(i)} - e_{p(j)} > +\frac{T}{2} \end{cases}.$$

Intuitively, if the difference between $e_{p(i)}$ and $e_{p(j)}$ is small, then the amount of translation should be equal to the difference. A simple example is $e_{p(i)} = e_{p(j)}$, where the aggressor and the victim have the same phase and aligned time zones. Thus, no phase translation is needed, instead of having $E_{p(j)p(i)} = T$ by the previous phase-shift operator. T will be involved in phase translation only when $e_{p(i)}$ and $e_{p(j)}$ are far away from each other.

Consider the example in Fig. 3 again with the new phase-shift operator. In this case, $E'_{p(u)p(v)} = -1$. Subtracting this phase-shift operator, the three aggressor ranges become $[-4, -2]$, [6, 8], and [16, 18], respectively. The previously missed coupling is now detected.

V. Circular Time Representation for Coupling Detection

Although coupling can be detected with $E'_{p(j)p(i)}$, the process of detection is complicated. Occurrences on three time zones need to be translated and checked. An important observation here is that if we translate the input windows of both the aggressor and the victim to the global time reference and take modulo T on them, coupling can be easily detected. For the example in Fig. 3, assuming that the global time reference coincides with v's local time zone, we have $[13, 15] \bmod T = [3, 5]$ for v, and $[-4, -2] \bmod T = [6, 8]$ for u. Given that $\tau_{u,v} = 1.01$, coupling is successfully detected. Theorem 1 establishes the correctness of this approach.

Theorem 1: Two vertices couple if and only if their switching windows overlap within a specified amount of time after being translated to the global time reference and taken modulo T.

Proof: ($\rightarrow$): If two vertices u and v couple, then, by the definition of coupling (assistive or opposing), it is possible that they switch simultaneously; i.e., their switching windows overlap within a specified amount of time. The fact of overlap will be preserved after being translated to the global time reference. In addition, since all the clock phases share a common period T, all switching windows are periodic with T, so are the overlaps among them. Therefore, the overlap between u and v will remain after being taken modulo T.

($\leftarrow$): On the other hand, since a clock schedule is periodic with a common period T, the fact that two switching windows overlap after being taken modulo T implies a coupling between them, which is independent on phase translations. ■

In addition, if switching windows at all the vertices are always computed with respect to the global time reference, phase translations are not necessary at all.

A formal treatment of taking modulo T on switching windows can be illustrated in a circular time representation. Fig. 4(a) shows the circular time representation of the three-phase clock schedule in Fig. 2. The whole circle represents the global time reference we selected. The time reference at the top point is 0. It increases clockwise until it goes back to the top, where reference T coincides with 0. Each phase i is then represented as an arc with endpoints s_i and e_i. Under this representation, each switching window becomes an arc. When propagating through a gate, an arc will be shifted clockwise and expanded, as shown in Fig. 4(b). The modulo T operation can be treated as a transformation from the original axis representation to the circular time representation to ensure that the values of the endpoints are within $[0, T)$, whenever an arc is shifted or expanded beyond the top point.

We must note that it is possible to have $A_v < a_v$ under the circular time representation. Therefore, $[a_v, A_v]$ is no longer interpreted as an interval in the original real axis but a *record* that indicates the starting and ending points of an arc. In the remainder of this paper, we will use "window" and "arc" interchangeably.

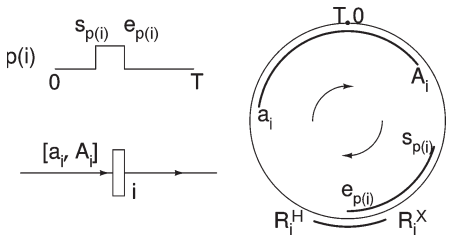


Fig. 5. Illustration of R_i^X and R_i^H for memory element i.

We then propose clock schedule verification with crosstalk under the circular time representation. It has two phases. In the first phase, we check if the clock schedule is valid (having no clocking violation) without crosstalk using the Szymanski and Shenoy algorithm. Given that we use (4) to characterize d_i instead of the more aggressive one in [20], it is guaranteed by [22] that the switching window without crosstalk is a subset of the switching window with crosstalk at each vertex. Therefore, a clock schedule is ensured to be invalid if it has clocking violations at this phase. If there is no clocking violation, (7)–(10) are carried out to obtain the switching windows at gates. We then translate the switching windows at memory elements and gates to the global time reference and take modulo T on them, i.e., represent them as arcs.

In the second phase, the occurrences of overlaps are used to update delays and switching arcs to take crosstalk effects into consideration. Whenever a switching arc is changed, the amount of change is propagated to its fan-outs and coupled wires. Propagation through a combinational vertex v can be expressed as

$$[d_v, D_v] = [(a_v + \delta_v) \bmod T, (A_v + \Delta_v) \bmod T].$$

Propagation through a latch i is formulated as

$$d_i = s_{p(i)}$$
$$D_i = \begin{cases} A_i, & \text{if } (A_i - a_i)\left(A_i - s_{p(i)}\right)\left(s_{p(i)} - a_i\right) > 0 \\ s_{p(i)}, & \text{otherwise} \end{cases}.$$

Propagation through a flop is formulated as

$$d_i = D_i = e_{p(i)}.$$

The change on one arc may trigger or cancel other succeeding couplings and result in more changes. The process of update is carried out over the whole circuit until convergence or a clocking violation is found.

For all $i \in V_L$, let

$$\left[R_i^X, R_i^H\right] \stackrel{\triangle}{=} \left[\left(e_{p(i)} - X_{p(i)}\right) \bmod T, \left(e_{p(i)} + H_{p(i)}\right) \bmod T\right]$$

as shown in Fig. 5.

Theorem 2 states the necessary and sufficient conditions for a valid clock schedule with crosstalk.

Theorem 2: Given that a clock schedule is valid without crosstalk, it is valid with crosstalk if and only if the input switching arc $[a_i, A_i]$ of each memory element $i \in V_L$ does not intersect with $\left[R_i^X, R_i^H\right]$ anytime before convergence.

Proof: Let $[\bar{a}_i, \bar{A}_i]$ denote the input switching arc at memory element i without crosstalk, $\forall i \in V_L$. Since the circuit is free of clocking violations under the clock schedule without crosstalk, $\bar{A}_i$ and $\bar{a}_i$ satisfy the setup and hold conditions (5) and (6), which implies that $[\bar{a}_i, \bar{A}_i]$ does not intersect with $[R_i^X, R_i^H]$.

Considering crosstalk, we have $[\bar{a}_i, \bar{A}_i] \subseteq [a_i, A_i]$ by [22]. If $[a_i, A_i]$ does not intersect with $[R_i^X, R_i^H]$ before convergence, $\forall i \in V_L$, then (5) and (6) are still kept with crosstalk. The clock schedule is still valid. On the other hand, if an intersection occurs at i before convergence, we must have either R_i^X or R_i^H (or both) on the arc $[a_i, A_i]$. The former stands for a hold violation, whereas the latter is a setup violation. The clock schedule is then invalid with crosstalk. ■

Based on this, a setup violation is caused when A_i is shifted clockwise to make the following inequality satisfied:

$$(A_i - a_i)\left(A_i - R_i^X\right)\left(R_i^X - a_i\right) > 0. \tag{11}$$

A hold violation is caused when a_i is shifted counterclockwise to make the following inequality satisfied:

$$(A_i - a_i)\left(A_i - R_i^H\right)\left(R_i^H - a_i\right) > 0. \tag{12}$$

Theorem 3 states that the convergence or a clocking violation will be found in polynomial time.

Theorem 3: The complexity of verifying clock schedule with crosstalk under the circular time representation is $O(|V_L|^3 + |E_C||E_I||V_L|)$.

Proof: Given a clock period, computing switching windows at memory elements by the Szymanski and Shenoy algorithm takes $O(|V_L|^3)$ time. If there is no clocking violation at this phase, i.e., (5) and (6) are satisfied, we then proceed to compute switching windows at gates. To do this, we will treat the outputs of memory elements as primary inputs and the inputs of memory elements as primary outputs and compute the switching windows at gates in their topological order, which takes $O(|E_I|)$ time. Translating all the switching windows to the global time reference and representing them under the circular time representation take $O(|V|)$ time.

To analyze the complexity of the second phase, we separate it into passes. At the beginning of each pass, we detect the occurrences of overlaps and update delays, which takes $O(|E_C|)$ time. Then, during the rest of the pass, we propagate the delay updates to other vertices without detecting overlap. The propagation through gates can be conducted in their topological order. It was shown in [18] that the propagation takes $O(|E_I||V_L|)$ time before convergence or a clocking violation is found. Given that the number of effective coupling capacitances increases every pass (otherwise, we stop with a converged solution), the number of passes is upper bounded by $O(|E_C|)$. Therefore, the complexity of the second phase is $O(|E_C||E_I||V_L|)$.

Summing up the complexities of the two phases, we obtain the complexity of verifying clock schedule with crosstalk under the circular time representation as $O(|V_L|^3 + |E_C||E_I||V_L|)$. ■

Compared with the methods in [3] and [4], our approach has two advantages. First, no phase translation is needed in the

second phase of our approach. Second, the overlap detection is much easier under the circular time representation. As a result, the process of verification is accelerated, which is confirmed by the experiments in Section VII.

VI. Algorithm for an Optimal Latch–Flop Configuration

A. Lower and Upper Bounds of a Feasible Clock Period

In [23], a set of constraints equivalent to (1)–(6) was proposed to characterize a feasible clock period, expressed as

$$e_{p(i)} \geq e_{p(j)} - w_{p(j)} + C^q_{j,i} - K^q_{j,i}T + X_{p(i)} \tag{13}$$

$$e_{p(i)} \leq e_{p(j)} - w_{p(j)} + c^q_{j,i} + \left(1 - K^q_{j,i}\right)T - H_{p(i)} \tag{14}$$

if j is a latch and

$$e_{p(i)} \geq e_{p(j)} + C^q_{j,i} - K^q_{j,i}T + X_{p(i)} \tag{15}$$

$$e_{p(i)} \leq e_{p(j)} + c^q_{j,i} + \left(1 - K^q_{j,i}\right)T - H_{p(i)} \tag{16}$$

if j is a flop. $C^q_{j,i}$ and $c^q_{j,i}$ are the maximum and minimum combinational delays, respectively, along any path q between j and another memory element i without crosstalk. Path q is combinational in (14) and (16); q could be sequential in (13) and (15); i.e., there could be memory elements on q. $K^q_{j,i}$ counts the number of occurrences that two consecutive memory elements x and y on q have $e_{p(x)} \geq e_{p(y)}$, recursively defined as

$$K^q_{j,k} = \begin{cases} 0, & \text{if } k = j \\ K^q_{j,b(k)}, & \text{if } e_{p(b(k))} < e_{p(k)} \\ K^q_{j,b(k)} + 1, & \text{if } e_{p(k)} \leq e_{p(b(k))} \end{cases} \tag{17}$$

where $b(k)$ represents the preceding memory element of k on q.

We use T_l to denote the minimum period that satisfies (13) for all q. Since e_i and w_i scale proportionally with T, we represent them as $e_i = \rho^e_i T$ and $w_i = \rho^w_i T$, where $0 \leq \rho^e_i, \rho^w_i < 1$. Then, (13) becomes

$$\left(K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j\right)T \geq C^q_{j,i} + X_{p(i)}.$$

By the definition of $K^q_{j,i}$ in (17), if $e_j < e_i$, then $K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j > 0$ since $K^q_{j,i} \geq 0$; if $e_j = e_i$, then $K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j \geq 1$ since $K^q_{j,i} \geq 1$; if $e_j > e_i$, then $K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j > 0$ since $K^q_{j,i} \geq 1$ and $\rho^e_i - \rho^e_j > -1$. Therefore, we have $K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j > 0$ for all q. As a result, T_l can be written as

$$T_l = \max_{q: j \rightsquigarrow i} \frac{C^q_{j,i} + X_{p(i)}}{K^q_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j}. \tag{18}$$

Lemma 1 states that T_l is a lower bound of a feasible clock period.

Lemma 1: The minimum period that can be possibly achieved with crosstalk is lower bounded by T_l.

Proof: Let q^+ be the path such that

$$\left(K^{q^+}_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j\right)T_l = C^{q^+}_{j,i} + X_{p(i)}.$$

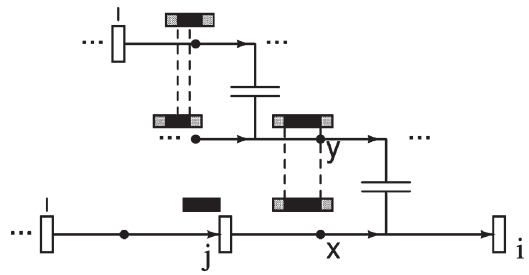


Fig. 6. Source graph of latch i.

Since the delays may be increased due to opposing couplings, the maximum delay along q^+ with crosstalk becomes $C'^{q^+}_{j,i} \geq C^{q^+}_{j,i}$. Therefore

$$\left(K^{q^+}_{j,i} + \rho^e_i - \rho^e_j\right)T_l \leq \left(K^{q^+}_{j,i} + \rho^e_i - \rho^e_j + \rho^w_j\right)T_l \leq C'^{q^+}_{j,i} + X_{p(i)}.$$

It implies that any value below T_l will cause a setup violation on q^+ with crosstalk independent of whether j and i are latches or flops. Hence, T_l is a lower bound of the solution. ■

To find an upper bound T_u of a feasible clock period, we use the minimum period that is feasible under the worst case coupling scenario (assuming all the couplings take effect). Both T_l and T_u can be computed by the algorithm in [23] in $O(|V_L|n^2 \log(|V_L|))$ time, where n is the number of clock phases.

Although an upper bound and a lower bound can be obtained, we cannot use binary search to find the optimal period. This is because the solution space may not be convex with crosstalk [3]. Therefore, we examine each candidate period from the lower bound incrementally. The first feasible period under which the clock schedule is valid with crosstalk is the optimal solution.

B. Heuristic

Given a clock period, we first scale the clock schedule proportionally and treat all the memory elements as latches to take advantage of time borrowing. Then, we compute the switching windows with crosstalk under the circular time representation. If there is no clocking violation, the clock period is feasible; otherwise, we apply a heuristic to remove the violations by replacing a subset of latches by flops.

Suppose a clocking violation occurs at $i \in V_L$. To find the latches for replacement, we recursively define the *source graph* of a vertex $v \in V$, or $sG(v)$, as

$$sG(v) = \begin{cases} \emptyset, & \text{if } v \in V_L \text{ and } d_v = D_v \\ sG(u) \cup \{u, (u,v)\}, & \text{if } (u,v) \in E_I \text{ or they couple} \end{cases}.$$

Then, $sG(i)$ can be obtained by tracing back from i along the interconnect edges and the coupling capacitances, where crosstalk takes effect, until we reach a memory element j with $d_j = D_j$ or a previously traversed vertex on each branch.

Fig. 6 shows an example of $sG(i)$, where vertical rectangles represent latches and horizontal rectangles represent the input switching windows of the vertices below them, with the gray parts indicating the crosstalk effects.

Theorem 4 states that the latches in $sG(i)$ are the candidates for replacement.

Theorem 4: At least one latch in $sG(i)$ has to be replaced with a flop to remove the clocking violation at i.

Proof: Suppose that the violation at i can be removed by only replacing some latches outside $sG(i)$. By the definition of $sG(i)$, there is no directed interconnect path from these latches to the vertices in $sG(i)$. It implies that the effect of the replacement on $sG(i)$ is to introduce more couplings, which cause more switching window expansions at the vertices in $sG(i)$. However, the violation at i cannot be removed by merely expanding the switching windows of the vertices in $sG(i)$. Therefore, some latch in $sG(i)$ has to be chosen for replacement. ■

If $sG(i)$ involves no coupling capacitance, the input switching window of i is determined by the minimum and maximum combinational delays from its critical predecessors. To remove a hold violation at i, a_i needs to be increased. This, by (2), implies that the earliest departure time at the earliest critical predecessor of i needs to be increased. Since the earliest departure time at the output of a latch is a constant by (4), the only way to increase it is to replace it by a flop. For a setup violation at i, we can trace back from i along the latest critical predecessors until we reach a memory element j in $sG(i)$ with $d_j = D_j$ such that A_i is determined by D_j through the maximum combinational delay from j to i. However, A_i cannot be decreased independent of whether j is a latch or a flop. In other words, the setup violation at i cannot be removed even without crosstalk effects. Therefore, the current clock period is infeasible.

Alternatively, if $sG(i)$ contains coupling capacitances, identifying the latches for replacement becomes very hard. Replacing a latch with a flop shrinks its output switching arc to a point. Unlike the situation without crosstalk, shrinking one switching arc may lead to decreases in maximum arrival times at the succeeding latches due to coupling misses. For the example in Fig. 6, when latch j is replaced by a flop, if the resulting switching window at x does not overlap with that at y and the maximum arrival time at x is smaller than the original D_x, then the maximum arrival time at latch i is actually decreased due to the replacement, which may help to fix the setup violation at i. In other words, the effect of a placement is dependent on the coupling configuration in the circuit, which in turn depends on the replacement. They are mutually dependent.

Therefore, we propose a heuristic that considers each latch in $sG(i)$ as a candidate. For each candidate, we estimate the effect of the replacement on the input switching window of i, assuming that the switching windows at other vertices will not change. Among the vertices whose individual replacements remove the violation, we choose the one whose latest arrival time is the closest to the falling edge of its clock phase. Intuitively, we want the increase of the latest arrival time of the latch being replaced to be as small as possible, so that the chance of introducing clocking violations at its succeeding memory elements is small. If the violation cannot be removed by replacing any latch in $sG(i)$, we choose the one that gives the least amount of violation. After replacing the chosen latch by a flop, we perform clock schedule verification with crosstalk on the new circuit. The above process is iteratively conducted until there is no clocking violation or the source graph has no candidate latch, for which we consider the current clock period infeasible. Since we replace a latch by a flop at each iteration and the number of latches in $sG(i)$ is no more than $|V_L|$, the number of iterations is upper bounded by $|V_L|$.

```
Algorithm
Input : A directed graph G = (V, E) and
        an n-phase clock schedule with period T.
Output: A configuration of mixed latches V − V_r
        and flops V_r with the minimum period T*.

Compute T_l and T_u;
T* ← T_l − ΔT;
Do
  Restore flops to latches;
  V_r ← ∅;
  T* ← T* + ΔT;
  Scale the clock schedule from T to T*;
  Do
    Compute switching windows w/o coupling under T*;
    If (5)-(6) are satisfied then
      Compute switching windows at gates by (7)-(10);
      Translate switching windows to global time;
      Take modulo T* on all the switching windows;
      Compute switching windows w/ coupling under T*;
      Check clocking violations by (11)-(12);
    If a clocking violation occurs on latch i then
      Choose latch j from sG(i) by heuristic;
      Replace j with a flop;
      V_r ← V_r ∪ {j};
  While (a new j is added into V_r);
While (clocking violations exist);
Return V_r and T*;
```

Fig. 7. Pseudocode of the algorithm.

Since switching windows at many vertices may vary when a latch is replaced by a flop, computing the effect of the replacement using the original switching windows may produce a result that is different from the later one by actually carrying out a verification on the new circuit. However, since we only allow one latch to be replaced at a time, it ensures that our assumption yields good estimations.

In Fig. 7, we give the pseudocode of the heuristic algorithm for finding the minimum clock period and the corresponding configuration of mixed latches and flops.

Theorem 5 gives the complexity of the algorithm.

Theorem 5: The algorithm in Fig. 7 terminates in $O(|V_L| \times n^2 \log(|V_L|) + |V_L|(|V_L|^3 + |E_C||E_I||V_L|)((T_u - T_l)/\Delta T))$ time with a feasible clock period.

Proof: It takes $O(|V_L|n^2 \log(|V_L|))$ to compute T_l and T_u by the algorithm in [23]. Computing the latch-to-latch minimum and maximum delays takes $O(|V_L|(|V_G| + |E_I|))$ time by the algorithm in [24].

Given a clock period, it takes $O(|V_L|^3 + |E_C||E_I||V_L|)$ time to verify it with crosstalk by Theorem 3. If there is a clocking violation at latch i, then finding $sG(i)$ and identifying the latch for replacement take $O(E)$ and $O(V_L V)$, respectively. Therefore, the complexity for checking the feasibility of a given period by our heuristic is $O(|V_L|(|V_L|^3 + |E_C||E_I||V_L|))$, where $O(|V_L|)$ bounds the number of iterations.

TABLE I
SEQUENTIAL CIRCUITS FROM ISCAS-89

Circuit	$\|V\|$	$\|E\|$	IO's	latches	gates
s386	197	397	14	24	159
s420	339	519	19	102	218
s838	691	1067	35	210	446
s1196	713	1250	28	156	529
s1488	752	1553	27	72	653
s3271	2156	3470	40	544	1572
s3330	2459	3691	113	557	1789
s3384	2443	3714	69	689	1685
s4863	2919	4911	65	512	2342
s5378	3707	5475	84	844	2779
s6669	4248	6852	138	1030	3080
s9234	6662	9666	75	990	5597
s13207	9460	13558	214	1295	7951
s15850	11718	16685	227	1719	9772
s35932	21627	35958	355	5207	16065

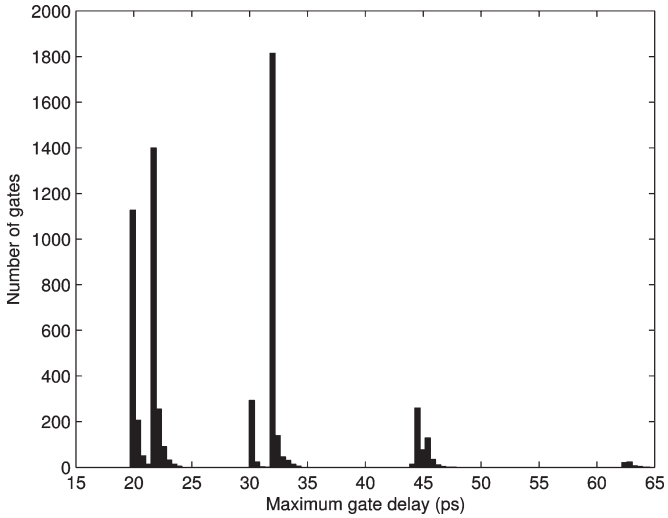


Fig. 8. Histogram of the maximum gate delays for c7552 in [25].

In conjunction with incremental search, the complexity of the algorithm is $O(|V_L|n^2\log(|V_L|)+|V_L|(|V_L|^3+|E_C||E_I||V_L|)((T_u-T_l)/\Delta T))$. In the worst case, the algorithm will return T_u, which is feasible. ■

VII. EXPERIMENTS

A. Experimental Setup

We implemented the algorithm in a personal computer with a 2.4-GHz Xeon CPU, a 512-kB second-level cache memory, and a 1-GB random access memory. Our test files were generated from ISCAS-89 sequential benchmark suite. They are summarized in Table I.

Due to the unavailability of extracted coupling information for ISCAS-89 circuits, we studied the extracted data of ISCAS-85 combinational circuits provided by [25] in 0.18-μm technology. For the largest circuit "c7552" in ISCAS-85 with 3513 gates, we illustrate the histogram of the maximum gate delays in Fig. 8, the histogram of the ratios of the minimum gate delay to the maximum in Fig. 9, the histogram of the interconnect delays in Fig. 10, the histogram of the interconnect parasitic capacitances in Fig. 11, and the histogram of the coupling capacitances in Fig. 12.

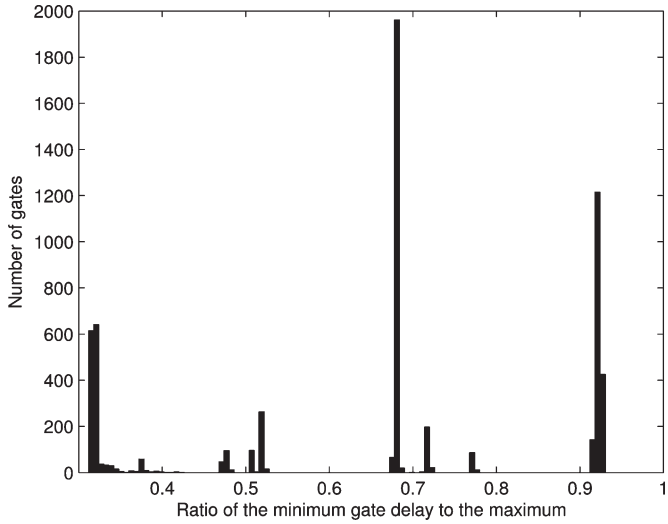


Fig. 9. Histogram of the ratios of the minimum gate delay to the maximum for c7552 in [25].

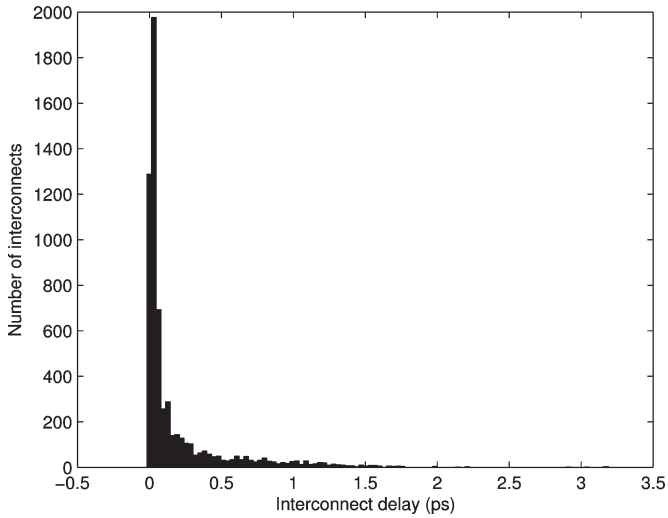


Fig. 10. Histogram of the interconnect delays for c7552 in [25].

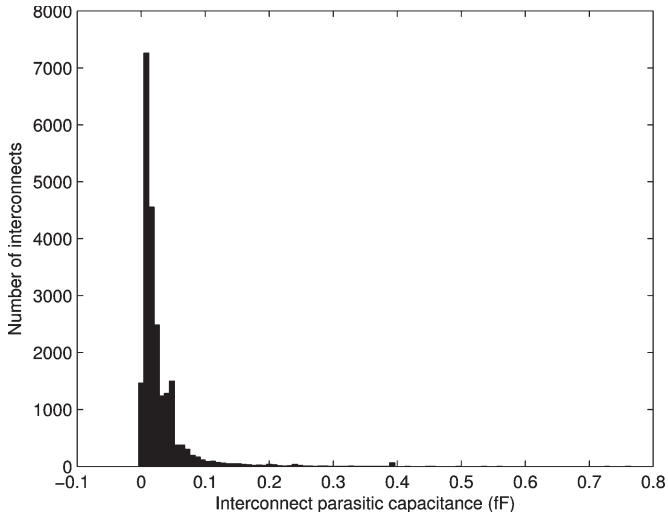


Fig. 11. Histogram of the interconnect parasitic capacitances for c7552 in [25].

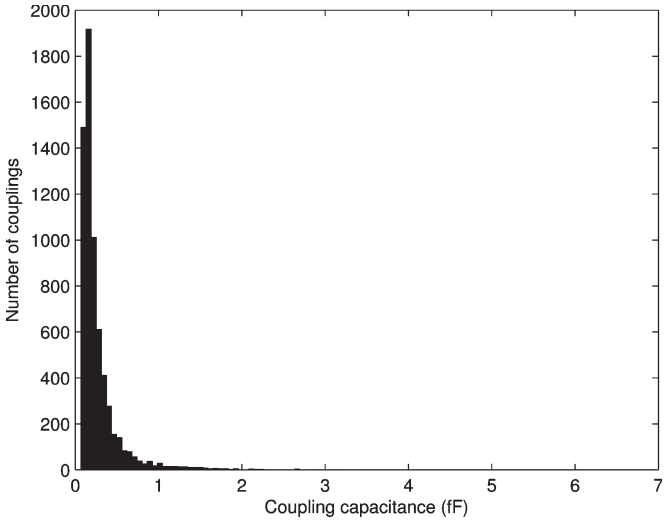


Fig. 12. Histogram of the coupling capacitances for c7552 in [25].

TABLE II
COMPUTED CLOCK PERIOD

Circuit	T_l	T_L^{opt}	T_{L+F}^{opt}	redu%	impr%	#veri	t(sec)
s386	33	38	33	100.0%	13.2%	9	0.00
s420	35	42	35	100.0%	16.7%	4	0.00
s838	49	62	49	100.0%	21.0%	38	0.04
s1196	74	89	89	0.0%	0.0%	769	0.68
s1488	72	76	75	25.0%	1.3%	36	0.04
s3271	94	116	97	86.4%	16.4%	59	0.30
s3330	143	196	143	100.0%	27.0%	8	0.02
s3384	102	120	120	100.0%	15.0%	14	0.09
s4863	155	170	170	0.0%	0.0%	97	0.71
s5378	162	220	162	100.0%	26.4%	2	0.02
s6669	158	209	164	88.2%	21.5%	86	1.49
s9234	254	344	254	100.0%	26.2%	41	1.52
s13207	543	573	553	66.7%	3.5%	80	4.04
s15850	446	654	654	0.0%	0.0%	4198	235.09
s35932	687	780	702	83.9%	10.0%	1263	198.04
avg				70.0%	13.2%		

We then assigned delays to ISCAS-89 sequential circuits based on the characteristics in Figs. 8–12. First of all, we observed that the maximum gate delays of c7552 are widely spread. So are the ratios of the minimum gate delay to the maximum. Note that it is not the absolute delay values but the ratios that are relevant. Therefore, we assigned each combinational vertex in the ISCAS-89 circuits with a random maximum delay within 0.5 and 2.5 (assuming no coupling capacitors); the minimum delay was randomly initialized with a value that is at most 0.5 less than the maximum delay. Random delay assignments expose our algorithm to various delay scenarios, which will result in a comprehensive evaluation of the algorithm. The effectiveness of our scheme may vary when applied to circuits with different gate delay distributions.

Next, we added coupling capacitors to the circuits. In c7552, the number of coupling capacitances is 6536, which is about half of the number of nodes, i.e., 11341. Therefore, we randomly added coupling capacitors equal in number to 50% of the vertices in each ISCAS-89 circuit. As a result, each vertex has a capacitor incident to it by average. On the other hand, the comparison between Figs. 11 and 12 implies that the coupling capacitance is generally $10\times$ the interconnect parasitic capacitance. Together with the interconnect delay histogram in Fig. 10, it can be expected that coupling induced delays are comparable to the gate delays. This is particularly true as the technology advances below 65 nm. Therefore, we assigned each coupling capacitor with a random delay between 0.0 and 2.0. After that, the delay of each coupling capacitor was added into the minimum and maximum delays of its two joint vertices.

Finally, we converted flops to back-to-back ϕ_1/ϕ_2 latches and used ASTRA, a min-period retiming tool developed by Sapatnekar and Deokar [24], to determine a symmetric nonoverlapping two-phase retiming. In Section VII-B, we report the obtained results based on the above setup.

B. Experimental Results

To find the optimal clock period, we search the space starting with the lower bound specified by T_l in (18), incrementing this period by 1 until we find a feasible period. We cannot use binary search because the solution space may not be convex. This is confirmed by our experiments that feasible periods may interleave with infeasible ones. For simplicity, we assume $\tau = 0$ (strict overlap) and zero setup and hold times. The results are reported in Table II. Column "T_l" lists the period lower bounds. Column "T_L^{opt}" lists the optimal periods for circuits with only latches. The optimal periods computed by our heuristic algorithm are listed in column "T_{L+F}^{opt}." Column "redu%" lists the percentage of period reduction with respect to the maximum possible reduction, i.e., $(T_L^{\text{opt}} - T_{L+F}^{\text{opt}})/(T_L^{\text{opt}} - T_l)$. Column "impr%" lists the percentage of period improvement with respect to the minimum period by using only latches, i.e., $(T_L^{\text{opt}} - T_{L+F}^{\text{opt}})/T_L^{\text{opt}}$. Since we choose the search step to be 1, the number of iterations from "T_l" to "T_{L+F}^{opt}" for each circuit is equal to their difference. Column "#veri" lists the number of clock schedule verifications that our algorithm has carried out for each circuit before T_{L+F}^{opt} is found. The runtime is reported in column "t(sec)" in seconds.

The results in Table II reveal three things. First, the values of T_l are the minimum periods we can possibly get for a symmetric two-phase clock schedule. However, crosstalk effects prevent the circuits from operating at T_l, which is illustrated by the fact that $T_L^{\text{opt}} > T_l$ for all the tested circuits. The overheads could be significant for some circuits. Second, by replacing a subset of latches into flops, our algorithm always successfully finds a feasible clock period T_{L+F}^{opt} with crosstalk. The effectiveness of our algorithm is shown by the circuits with $T_{L+F}^{\text{opt}} < T_L^{\text{opt}}$. In fact, half of the circuits are able to run at the indicated period lower bounds after the replacement, i.e., $T_{L+F}^{\text{opt}} = T_l$. For these circuits, our algorithm achieves 100% period reduction. On average, the percentage of period reduction is 70.0%, and the percentage of period improvement is 13.2%. Last, our algorithm is efficient. For the longest case "s15850" with more than 11 000 vertices, the runtime 235.09 s are measured for 208 iterations and overall 4198 clock schedule verifications with crosstalk, i.e., 0.056 s per verification. Given that the approaches in [3] and [4] take more than 30 s (scaled based on the difference in CPU frequency between their machines and ours) to do a single clock schedule verification with crosstalk for circuits with around 4000 vertices, the proposed

TABLE III
CHANGE OF CONTRIBUTING CAPACITORS

Circuit	total cap	cap_{L}	$\text{cap}_{\text{L+F}}$	flop	flop/$\lvert V\rvert$
s386	79	54	42	8	4.06%
s420	109	32	1	3	0.88%
s838	223	136	18	37	5.35%
s1488	326	279	13	11	1.46%
s3271	786	503	6	48	2.23%
s3330	894	104	27	7	0.28%
s3384	842	610	595	13	0.53%
s5378	1389	908	190	1	0.03%
s6669	1540	1051	12	30	0.71%
s9234	2798	1714	231	40	0.60%
s13207	3975	3412	249	23	0.24%
s35932	8032	4296	2760	98	0.45%

approach under the circular time representation is much more efficient. The efficiency is due to the proposed efficient coupling detection. In addition, the efficiency is independent on delay and coupling capacitance scenarios since coupling detection is a generic operation in clock schedule verification. In other words, the running time improvement over [3] and [4] will be similar even when extracted coupling data are available.

For the circuits with $T_{L+F}^{\text{opt}} < T_{L}^{\text{opt}}$, we list in Table III the number of contributing capacitors that affect switching windows before and after the replacement in columns "cap_L" and "cap_{L+F}," respectively. The number of latches chosen for replacement in each circuit under T_{L+F}^{opt} is listed in column "flop." The ratio between the number of flops and the number of vertices is listed in column "flop/$|V|$." We can see from Table III that "cap_{L+F}" is less than cap_L for all circuits. It means that the crosstalk effects are reduced after the replacement. In addition, the "flop/$|V|$" ratio is small. It implies that if a flop is implemented as two back-to-back latches, the area overhead due to the replacement is negligible.

To illustrate the crosstalk effects on clock period when the number of coupling capacitors varies, we use "s9234" as an example. The results are presented in Fig. 13, where the x-axis is the percentage of the number of capacitors with respect to the number of gates, and the y-axis is the clock period. The results confirm that the overhead due to crosstalk effects, i.e., $(T_{L}^{\text{opt}} - T_l)/T_l$, grows severer as the number of capacitors increases. On the other hand, we find that the effectiveness of our algorithm is hardly affected by the number of capacitors. It achieves 100% period reduction, i.e., $T_{L+F}^{\text{opt}} = T_l$, for almost all different capacitor numbers. As a result, the percentage of period improvement increases when more capacitors are present in the circuit.

We also notice that there are a few circuits in Table II that our algorithm cannot improve. It happens when replacing a latch by a flop in a circuit causes clocking violations at its succeeding memory elements. Recall that the replacement of a latch not only shrinks the width of the switching window but also increases the window to the falling edge of its clock phase. Intuitively, if we can adjust the falling edge of the clock phase after the replacement, the violations at the succeeding memory elements may be avoided. The technique for adjusting clock phases is also known as *clock skew scheduling* [26]. We will consider incorporating it into the proposed algorithm as our future work.

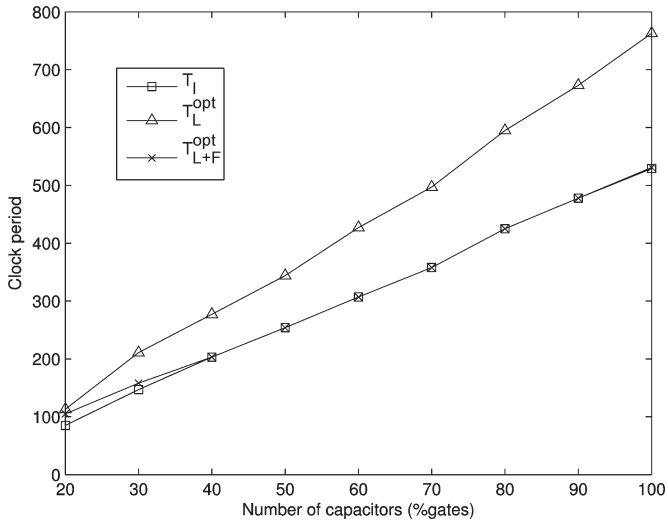


Fig. 13. Crosstalk effects on clock period as the number of capacitors varies for s9234.

VIII. CONCLUSION

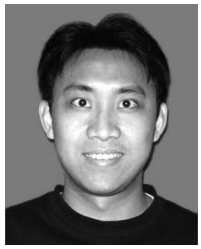
The tradeoff between a latch and a flop in sequential circuit designs with crosstalk is formulated as seeking a configuration of mixed latches and flops to minimize the clock period. A circular time representation is proposed for coupling detection without additional phase translations, which are otherwise required in state-of-the-art approaches [3], [4]. We show that clock schedule verification under the circular time representation is easier. A heuristic algorithm is presented for finding the optimal configuration of latches and flops. Experimental results show that our algorithm is both effective and efficient.

The proposed framework and solution approach can equally apply to crosstalk induced by capacitive and inductive couplings.

REFERENCES

[1] C. Ebeling and B. Lockyear, "On the performance of level-clocked circuits," in *Proc. Adv. Res. VLSI*, 1995, pp. 342–356.
[2] Semiconductor Industry Association. (2001). *International Technology Roadmap for Semiconductors*. [Online]. Available: http://public.itrs.net
[3] S. Hassoun, C. Cromer, and E. C.-Gamez, "Static timing analysis for level-clocked circuits in the presence of crosstalk," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 9, pp. 1270–1277, Sep. 2003.
[4] H. Zhou, "Clock schedule verification with crosstalk," in *Proc. ACM Int. Workshop Timing Issues Specification and Synthesis Digital Syst.*, 2002, pp. 78–83.
[5] S. Hassoun, "Critical path analysis using a dynamically bounded delay model," in *Proc. Des. Autom. Conf.*, Los Angeles, CA, Jun. 2000, pp. 260–265.
[6] P. F. Tehrani, S. W. Chyou, and U. Ekambaram, "Deep sub-micron static timing analysis in presence of crosstalk," in *Proc. Int. Symp. Quality Electron. Des.*, 2000, pp. 505–512.
[7] R. Levy, D. Blaauw, G. Braca, A. Dasgupta, A. Grinshpon, C. Oh, B. Orshav, S. Sirichotiyakul, and V. Zolotov, "Clarinet: A noise analysis tool for deep submicron design," in *Proc. Des. Autom. Conf.*, 2000, pp. 233–238.
[8] R. Arunachalam, K. Rajagopal, and L. T. Pilleggi, "Taco: Timing analysis with coupling," in *Proc. Des. Autom. Conf.*, Los Angeles, CA, Jun. 2000, pp. 266–269.
[9] S. S. Sapatnekar, "A timing model incorporating the effect of crosstalk on delay and its application to optimal channel routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 5, pp. 550–559, May 2000.

[10] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou, "Optimizing two-phase, level-clocked circuitry," *J. ACM*, vol. 44, no. 1, pp. 148–199, Jan. 1997.
[11] P. Chen, D. A. Kirkpatrick, and K. Keutzer, "Switching window computation for static timing analysis in presence of crosstalk noise," in *Proc. Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 2000, pp. 331–337.
[12] R. Arunachalam, R. D. Blanton, and L. T. Pileggi, "False coupling interactions in static timing analysis," in *Proc. Des. Autom. Conf.*, 2001, pp. 726–731.
[13] P. Chen and K. Keutzer, "Toward true crosstalk noise analysis," in *Proc. Int. Conf. Comput.-Aided Des.*, 1999, pp. 132–137.
[14] T. Xiao and M. Marek-Sadowska, "Functional correlation analysis in crosstalk induced critical paths identification," in *Proc. Des. Autom. Conf.*, 2001, pp. 653–656.
[15] F. Dartu and L. T. Pileggi, "Calculating worst-case gate delays due to dominant capacitance coupling," in *Proc. Des. Autom. Conf.*, Anaheim, CA, Jun. 1997, pp. 46–51.
[16] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of worst-case aggressor alignment for delay calculation," in *Proc. Int. Conf. Comput.-Aided Des.*, San Jose, CA, Nov. 1998, pp. 212–219.
[17] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo, "Driver modeling and alignment for worst-case delay noise," in *Proc. Des. Autom. Conf.*, 2001, pp. 720–725.
[18] T. G. Szymanski and N. Shenoy, "Verifying clock schedules," in *Proc. Int. Conf. Comput.-Aided Des.*, 1992, pp. 124–131.
[19] N. V. Shenoy, "Timing issues in sequential circuits," Ph.D. dissertation, Univ. California Berkeley, CA, 1993.
[20] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "checkT_c and mint$_c$: Timing verification and optimal clocking of synchronous digital circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 1990, pp. 552–555.
[21] T. G. Szymanski, "Computing optimal clock schedules," in *Proc. Des. Autom. Conf.*, 1992, pp. 399–404.
[22] H. Zhou, "Timing analysis with crosstalk is a fixpoint on a complete lattice," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 9, pp. 1261–1269, Sep. 2003.
[23] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Graph algorithms for clock schedule optimization," in *Proc. Int. Conf. Comput.-Aided Des.*, 1992, pp. 132–136.
[24] S. S. Sapatnekar and R. B. Deokar, "Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 10, pp. 1237–1248, Oct. 1996.
[25] X. Lu and W. Shi, *Layout and Parasitic Information for Iscas Circuits*, College Station, TX: Texas A&M Unversity. [Online]. Available: http://dropzone.tamu.edu/~xiang/iscas.html
[26] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 945–951, Jul. 1990.



Chuan Lin received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 2002, and the Ph.D. degree in computer engineering from Northwestern University, Evanston, IL, in 2006.

He is a member of Technical Staff with Magma Design Automation Inc., San Jose, CA. His research interests are on VLSI computer-aided design, especially deep-submicrometer physical design.

Hai Zhou (M'04–SM'04) received the B.S. and M.S. degrees in computer science and technology from Tsinghua University, Bejing, China, in 1992 and 1994, respectively, and the Ph.D. degree in computer sciences from the University of Texas, Austin, in 1999.

He was with the Advanced Technology Group, Synopsys, Inc., Mountain View, CA. He then joined the faculty of the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, where he is currently an Assistant Professor of electrical engineering and computer science. His research interests include very large scale integrated computer-aided design, algorithm design, and formal methods.

Dr. Zhou has served on the technical program committees of many conferences on very large scale integrated circuits and computer-aided design. He was a recipient of the CAREER Award from the National Science Foundation in 2003.