Trade-off between Latch and Flop for Min-Period Sequential Circuit Designs with Crosstalk*

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

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 trade-off 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.

1 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: edge-triggered *flops* store the data when the clock switches; 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 [7].

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 set-up 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 set-up and hold conditions on paths between any two latches need to be considered. These conditions become even more complex in the presence of a multi-phase clock schedule.

On the other hand, with increasing clock frequencies and shrinking process geometries in deep sub-micron technology, both capacitive and inductive coupling (also known as crosstalk) 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 [15]. Besides introducing noises on quiet wires, crosstalk may greatly change the wire delays, hence affect the timing analysis. If an aggressor and a victim switch simultaneously in the same direction, the victim will speed up, called *assistive coupling*. Likewise, if an aggressor and a victim switch simultaneously in opposite directions, the victim will slow down, 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 the paper.

The problem of clock schedule verification without considering crosstalk has been elegantly solved by Szymanski and Shenoy [20, 17]. Their approach was also extended to incorporate crosstalk [10, 22], where coupling detection required additional phase translations and comparisons. Several techniques have been proposed to evaluate crosstalk effects on combinational delays. Some are based on iterative techniques [2, 13]; some are based on the propagation of events [4]; others are based on more complex mathematical formulations [9]. Consideration of the functional correlation of the victim and the aggressors allows further accuracy in analysis [1, 3, 21]. 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 [6, 8, 18]. 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.

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 details in Section 2. 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. Firstly, we

*This work is supported by the NSF under CCR-0238484.

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 [10, 22]. Furthermore, since complicated phase translations are avoided, clock schedule verification under the circular time representation is more efficient. Secondly, we formulate the trade-off 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 fast heuristic algorithm to solve this problem. Experiments showed promising results. Although our implementation is based on dynamically bounded delay model proposed by Hassoun [9], the framework and solution proposed here can be easily extended to utilize other discrete coupling models.

The rest of this paper is organized as follows. Section 2 presents the motivation and problem formulation. Section 3 describes the models we will use in this paper. Section 4 recaps the previous work. The circular time representation is proposed in Section 5. Our algorithm is presented in Section 6, followed by experimental results in Section 7. Conclusions are given in Section 8.

2 Motivation and problem formulation

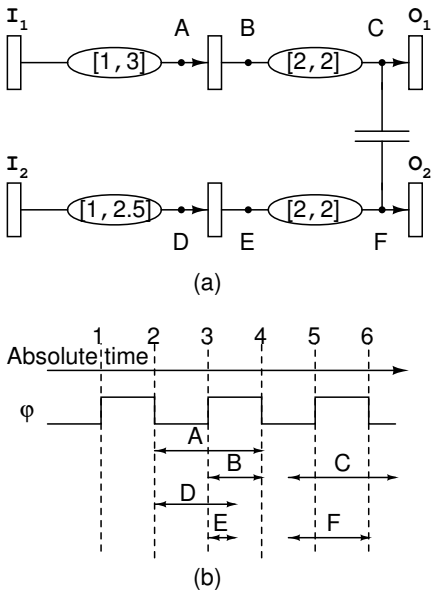

Figure 1: Crosstalk effects on system performance.

We now use an example in Figure 1(a) to show that the amount of timing uncertainty due to crosstalk accumulated through the 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 Figure 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 while assistive coupling decreases the delay by 0.5. Suppose that the set-up 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, illustrated in Figure 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 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 set-up 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 set-up violation at O_1 is now removed. The new circuit can still work under period 2.

This example shows that the trade-off 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 the clocking conditions with crosstalk under the minimal clock period.

3 Models and notations

3.1 Clock model

A *clock schedule* for a circuit is a set of periodical signals $\phi_1, ..., \phi_n$ with width w_i of the phase ϕ_i and a common period T. A three phase clock schedule is shown in Figure 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 $e_i = \frac{i}{n}T$, the clock schedule is *symmetric* [11]. The set-up 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.

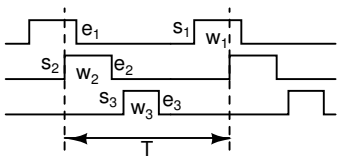

Figure 2: Three phase clock schedule with period T.

3.2 Delay model

We choose to use the dynamically bounded delay model [9] to capture the delay variations due to crosstalk. More specifically, each vertex v has a combinational delay range $[\delta_v, \Delta_v]$.

In addition, for each coupling capacitance between v and an aggressor u, we associate it with five parameters: $\delta_{u,v}$, $\delta_{v,u}$, $\Delta_{u,v}$, $\Delta_{v,u}$ and $\tau_{u,v}$. 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 δ_v is decreased by $\delta_{u,v}$ and Δ_v is increased by $\Delta_{u,v}$ *if they have never been done so.*

While more accurate continuous models are possible, e.g., [5], the chosen model is a generalization of discrete coupling models, such as ones that assume a 0 X, 1 X, or 2 X effective coupling capacitance, e.g., [13]. The framework and solution proposed here can be easily extended to utilize other discrete coupling models.

3.3 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: the gates V_G and the memory elements V_L, and $E = E_I \cup E_C$ is the union of two sets of edges: the interconnects E_I and the coupling capacitances E_C. Vertices and coupling capacitances are modeled by dynamically bounded delay model. Interconnect delays are available since we are considering circuits after placement and routing. We assume that all the 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 $p(v)$. However, to simplify the presentation, we only talk about latches and the flops that operate at the falling edges. The proposed approach can be easily extended to include the flops that operate at the rising edges. In the remainder of this paper, we simply use flops to refer to those that operate at the falling edges.

4 Previous work

The problem of clock schedule verification without considering crosstalk has been elegantly solved by Szymanski and Shenoy [20, 17]. 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 fanins through shortest and longest path delays to i. Timing conditions for latches are formulated as [17]

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

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

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

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

$$A_i \leq T - X_i \quad (5)$$

$$a_i \geq H_i \quad (6)$$

where $C_{j,i}$ and $c_{j,i}$ represent the maximal and minimal combinational delays from latch j to latch i respectively, and $E_{p(j)p(i)}$ is a phase shift operator defined as [12]

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

Note that used here are local times referring to local time zones that end with the phase falling edges. Timing equations involving flops can be similarly derived. 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 [12]. This is because [19] 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 which 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.* [10] proposed to assign each gate v a *virtual* phase, also denoted as $p(v)$, which can be derived by analyzing the phases of the memory elements in the combinational fanin and fanout of the gate. Based on the virtual phases, the switching windows at combinational gates can be computed in a similar fashion as those at memory elements, expressed as

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

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

$$d_v = a_v + \delta_v \quad (9)$$

$$D_v = A_v + \Delta_v \quad (10)$$

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 [22] 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 approaches was not easy. [10] 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 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. It was shown 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 Figure 3 (taken from [10]), 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: the previous $[5 - T, 7 - T]$, the current $[5, 7]$ and the following $[5 + T, 7 + T]$, then the ranges, respectively, become $[-14, -12]$, $[-4, -2]$, and $[6, 8]$. 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

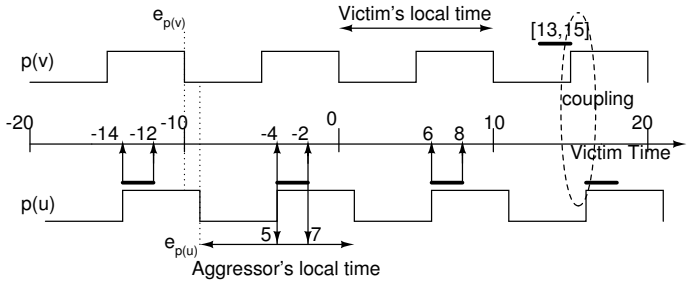

Figure 3: An example where coupling is missed by $E_{p(u)p(v)}$.

$E'_{p(j)p(i)}$, defined as

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

Consider the example in Figure 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 now become $[-4, -2]$, $[6, 8]$, and $[16, 18]$ respectively. The previously missed coupling is now detected.

5 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 Figure 3, assuming that the global time reference coincides with v's local time zone, we have $[13, 15] \mod T = [3, 5]$ for v, and $[-4, -2] \mod T = [6, 8]$ for u. Given that $\tau_{u,v} = 1.01$, coupling is successfully detected. The following theorem 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 and taken modulo T.

($\leftarrow$): On the other hand, since a clock schedule is periodical, 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. Figure 4(a) shows the circular time representation of the three phase clock schedule in Figure 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 Figure 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.

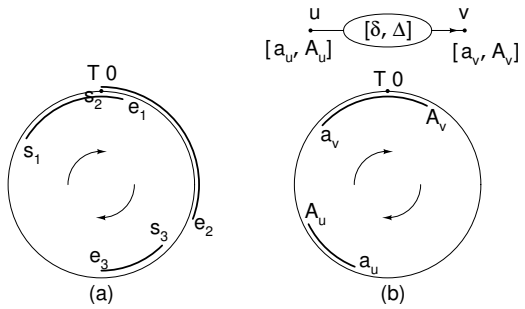

Figure 4: (a) The circular time representation of the clock schedule in Figure 2. (b) Switching window propagation through a combinational vertex under the circular time representation.

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 are using (4) to characterize d_i instead of the aggressive one, the switching window without crosstalk is guaranteed to be a subset of the switching window with crosstalk at each vertex [23]. 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 window at each gate. We then translate switching windows at all the memory elements and gates to the global time reference, and take modulo T on them, that is, 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 all its fanouts and coupled wires. Propagation through a combinational vertex v can be expressed as

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

Propagation through a latch i is formulated as

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

Propagation through a flop can be similarly formulated. 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

$$[R_i^X, R_i^H] \triangleq [(e_{p(i)} - X_{p(i)}) \text{mod } T,\ (e_{p(i)} + H_{p(i)}) \text{mod } T],$$

as shown in Figure 5.

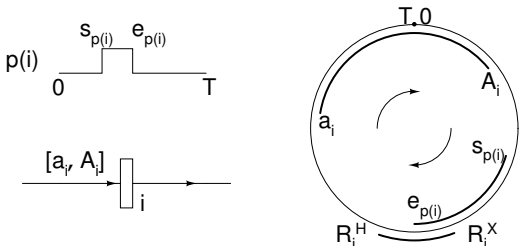

Figure 5: Illustration of R_i^X and R_i^H for memory element i.

The following theorem states the necessary and sufficient conditions for a valid clock schedule with crosstalk.

Theorem 2 *Given that the 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 $[R_i^X, R_i^H]$ anytime before convergence.*

Proof: Let $[\bar{a}_i, \bar{A}_i]$ denote the input switching arc of 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 (5) and (6) respectively, which implies that $[\bar{a}_i, \bar{A}_i]$ does not intersect with $[R_i^X, R_i^H]$.

According to [23], $[\bar{a}_i, \bar{A}_i] \subseteq [a_i, A_i]$. If $[a_i, A_i]$ does not intersect with $[R_i^X, R_i^H]$ before convergence, $\forall i \in V_L$, then (5)-(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 while the latter is a set-up violation. The clock schedule is then invalid with crosstalk. □

Based on this, a set-up violation is caused when A_i is shifted clockwise to make the following inequality satisfied.

$$(A_i - a_i)(A_i - R_i^X)(R_i^X - a_i) \ > \ 0 \qquad (11)$$

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

$$(A_i - a_i)(A_i - R_i^H)(R_i^H - a_i) \ > \ 0 \qquad (12)$$

Compared with the methods in [10, 22], our approach has two advantages. Firstly, no phase translation is needed in the second phase of verifying the clock schedule with crosstalk. Secondly, 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.

6 Algorithm for an optimal latch-flop configuration

6.1 Lower and upper bounds of a feasible clock period

In [16], a set of constraints equivalent to (1)-(6) was proposed to characterize a feasible clock period, expressed as

$$e_{p(i)} \ \geq \ e_{p(j)} - w_{p(j)} + C_{j,i}^q - K_{j,i}^q T + X_{p(i)} \qquad (13)$$
$$e_{p(i)} \ \leq \ e_{p(j)} - w_{p(j)} + c_{j,i}^q + (1 - K_{j,i}^q)T - H_{p(i)} \qquad (14)$$

if j is a latch, and

$$e_{p(i)} \ \geq \ e_{p(j)} + C_{j,i}^q - K_{j,i}^q T + X_{p(i)} \qquad (15)$$
$$e_{p(i)} \ \leq \ e_{p(j)} + c_{j,i}^q + (1 - K_{j,i}^q)T - H_{p(i)} \qquad (16)$$

if j is a flop, where $C_{j,i}^q$ and $c_{j,i}^q$ are the maximal and minimal combinational delays along any combinational path q between j and another memory element i without crosstalk. $K_{j,i}^q$ 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

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

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

The lower and upper bounds of a feasible clock period are given by the following theorem.

Theorem 3 *Let T_l denote the minimal period that satisfies (13) for all q, and let T_u denote the maximal period that satisfies (16) for all q, where $q = j \rightsquigarrow i$, $\forall j, i \in V_L$. If $0 < T_l \leq T_u < \infty$, then the minimal period that can be possibly achieved with crosstalk is within interval $[T_l, T_u]$.*

Proof: Since e_i and w_i scale proportionally with T, we represent them as $e_i = \rho_i^e T$ and $w_i = \rho_i^w T$, where $0 \leq \rho_i^e, \rho_i^w < 1$. Then (13) becomes

$$(K_{j,i}^q + \rho_i^e - \rho_j^e + \rho_j^w)T \geq C_{j,i}^q + X_{p(i)}.$$

Since $T_l > 0$, we have $K_{j,i}^q + \rho_i^e - \rho_j^e + \rho_j^w > 0$ for all q, and there exists a particular path q^+ such that

$$(K_{j,i}^{q^+} + \rho_i^e - \rho_j^e + \rho_j^w)T_l = C_{j,i}^{q^+} + X_{p(i)}.$$

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

$$\begin{aligned}(K_{j,i}^{q^+} + \rho_i^e - \rho_j^e)T_l \ &\leq \ (K_{j,i}^{q^+} + \rho_i^e - \rho_j^e + \rho_j^w)T_l \\ &\leq \ C_{j,i}^{'q^+} + X_{p(i)}.\end{aligned}$$

It implies that any value below T_l will cause a set-up 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.

On the other hand, (16) has the following form

$$(K_{j,i}^q - 1 + \rho_i^e - \rho_j^e)T \leq c_{j,i}^q - H_{p(i)}.$$

Since $T_u < \infty$, we know that T_u is determined on some particular path q^- such that $K_{j,i}^{q^-} - 1 + \rho_i^e - \rho_j^e > 0$ and

$$(K_{j,i}^{q^-} - 1 + \rho_i^e - \rho_j^e)T_u = c_{j,i}^{q^-} - H_{p(i)}.$$

Likewise, if crosstalk is considered, the minimal delay along q^- becomes $c_{j,i}^{'q^-} \leq c_{j,i}^{q^-}$. Therefore,

$$\begin{aligned}(K_{j,i}^{q^-} - 1 + \rho_i^e - \rho_j^e + \rho_j^w)T_u \ &\geq \ (K_{j,i}^{q^-} - 1 + \rho_i^e - \rho_j^e)T_u \\ &\geq \ c_{j,i}^{'q^-} - H_{p(i)}.\end{aligned}$$

Independent of whether j and i are latches or flops, any

value beyond T_u will cause a hold violation on q^-. Hence, T_u is an upper bound of the solution. □

The lower and upper bounds can be easily computed once $c^q_{j,i}$, $C^q_{j,i}$ and $K^q_{j,i}$ are obtained, which takes $O\big(V_L(V_G + E_I)\big)$ time by the Szymanski and Shenoy algorithm. If the computed lower bound is actually negative or ∞, it implies a design error in assigning phases to the memory elements in the circuit. If the computed upper bound is ∞, it means that the circuit is immune from hold violations no matter what the clock period is and how the configuration is chosen. For this case, we can use the period that is feasible under the worst case coupling scenario (assuming all the couplings take effect) as the upper bound.

Although an upper 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 [10]. 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.

6.2 A 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 $i \in V_L$. In order 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. Figure 6 shows an example of sG, 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.

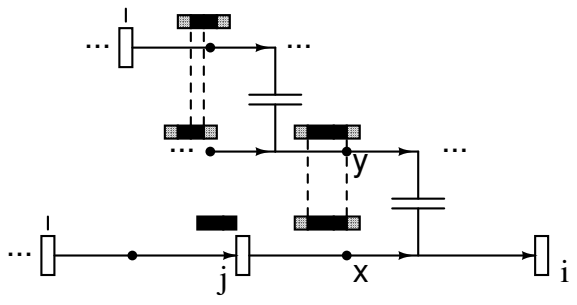

Figure 6: The source graph of latch i.

The following theorem states that the latches in sG are the candidates for replacement.

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

Proof: Suppose 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 minimal and maximal 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 set-up violation at i, we can trace back from i along the latest critical predecessors until we reach a root j in $sG(i)$ with $d_j = D_j$, such that A_i is determined by D_j through the maximal combinational delay from j to i. However, A_i cannot be decreased by replacing j with a flop. In other words, the set-up 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 Figure 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 set-up 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 its phase falling edge. 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 other 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 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 Figure 7, we give the algorithm for finding the minimal clock period and the corresponding configuration of mixed

latches and flops.

Algorithm Optimal latch-flop configuration
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 minimal 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*;
```

Figure 7: Pseudocode of the algorithm

The following theorem gives the complexity of the algorithm.

Theorem 5 *The algorithm in Figure 7 terminates with a solution in* $O\big((V_L E_I + V_L^3 + E_C E_I + V_L V_G)\frac{T_u - T_l}{\Delta T}\big)$ *time.*

Proof: Computing the latch-to-latch minimum and maximum delays for the Szymanski and Shenoy algorithm takes $O\big(V_L(V_G + E_I)\big)$ time, which is also the time for computing T_l and T_u. Their algorithm runs in $O(V_L^3)$. The run-time for computing switching windows at gates and representing them under the circular time representation take $O(E_I)$ and $O(V)$ time respectively. The complexity of verifying the clock schedule with crosstalk under the circular time representation is $O(E_C E)$. If there is a clocking violation at latch i, finding $sG(i)$ and identifying the latch for replacement take at most $O(E)$ and $O(V_L V)$ respectively. Therefore, the overall complexity is $O\big((V_L E_I + V_L^3 + E_C E_I + V_L V_G)\frac{T_u - T_l}{\Delta T}\big)$. In the worst case, the algorithm will return T_u, which is feasible. □

7 Experimental results

We implemented the algorithm in a PC with a 2.4 GHz Xeon CPU, 512 KB 2nd level cache memory and 1GB RAM. Our test files were generated from ISCAS-89 benchmark suite. Each combinational vertex in the circuit was randomly assigned with a maximum delay within 0.5 and 2.5; the minimum delay was randomly initialized with a value that is at most 0.5 less than the maximum delay. We randomly added capacitors equal in number to 10% of the total circuit vertices. Each capacitor was randomly assigned a delay between 0.0 and 3.0. The delay was also added into the minimum and maximum delays of the two joint vertices. We converted flops to back-to-back ϕ_1/ϕ_2 latches and used ASTRA, a min-period retiming tool by Sapatnekar and Deokar [14], to determine a symmetric, non-overlapping, two-phase retiming. The circuits used are summarized in Table 1.

Table 1: Sequential circuits from ISCAS-89

Circuit	$\|V\|$	$\|E\|$	IO's	latches	gates
s386	197	397	14	24	159
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
s9234	6662	9666	75	990	5597
s15850	11718	16685	227	1719	9772
s35932	21627	35958	355	5207	16065

To find the optimal clock period, we search the space starting with the lower bound specified by T_l in Theorem 3, incrementing this period by 0.01 until we find a feasible period. We cannot use a binary search here 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 set-up and hold times. The results are reported in Table 2. 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 algorithm are listed in Column "T_{L+F}^{opt}". The effectiveness of our algorithm is illustrated by the highlighted cases where $T_L^{\text{opt}} > T_{L+F}^{\text{opt}}$. Column "impr%" lists the reduction achieved with respect to the maximum possible reduction (i.e. $T_L^{\text{opt}} - T_l$). Since we choose the search step to be 0.01, the number of iterations from "T_l" to "T_{L+F}^{opt}" for each circuit is their difference multiplied by 100. Column "#veri" lists the number of clock schedule verifications the algorithm has carried out for each circuit before T_{L+F}^{opt} is found. The runtime is reported in column "t(sec)" in seconds.

Table 2: Optimal clock period

Circuit	T_l	T_L^{opt}	T_{L+F}^{opt}	impr%	#veri	t(sec)
s386	21.84	24.08	24.08	0.0%	1228	0.31
s838	34.28	**35.78**	**34.28**	100.0%	10	0.00
s1196	53.48	**55.63**	**55.39**	11.2%	7275	3.87
s1488	34.07	**35.83**	**35.54**	16.5%	398	0.25
s3271	40.31	**44.20**	**43.95**	6.4%	3494	8.85
s3330	34.78	**36.33**	**34.78**	100.0%	9	0.02
s3384	56.62	**58.43**	**56.62**	100.0%	16	0.04
s4863	61.24	**63.00**	**61.60**	79.5%	300	1.54
s5378	57.61	**61.97**	**57.61**	100.0%	2	0.00
s9234	80.53	**84.13**	**80.53**	100.0%	7	0.12
s15850	129.64	**133.71**	**129.64**	100.0%	5	0.19
s35932	77.88	**84.36**	**82.23**	32.9%	2178	62.04

The results in Table 2 reveal three things. Firstly, the values of T_l are the minimal 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 example, 10.3% for "s386". It will be severer when more and bigger coupling capacitors are present in the circuit. Secondly, by replacing a subset of latches into flops, our algorithm always successfully finds a feasible clock period T_{L+F}^{opt} with crosstalk. In fact, our algorithm is so effective that 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. Lastly, our algorithm is efficient. Most circuits can be processed within seconds. For the longest case "s35932" with over 21,000 vertices, the runtime 62.04 seconds are measured for 436 iterations and overall 2178 clock schedule verifications with crosstalk. Given that the approaches in [10, 22] take over 30 seconds (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 4,000 vertices, the proposed approach under the circular time representation is much more efficient. The efficiency is due to the proposed efficient coupling detection.

In Table 3, we list the number of contributing capacitors that affect switching windows in the circuit before and after the replacement in column "c.cap$_{\text{L}}$" and column "c.cap$_{\text{L+F}}$", respectively. The number of latches chosen for replacement in each circuit under T_{L+F}^{opt} is listed in column "flop". Note that "c.cap$_{\text{L+F}}$" is not necessarily less than "c.cap$_{\text{L}}$". Even if they are equal, the constituent contributing capacitors may be different.

Table 3: Change of contributing capacitors

Circuit	total cap	c.cap$_{\text{L}}$	c.cap$_{\text{L+F}}$	flop
s386	19	13	13	0
s838	69	32	3	9
s1196	71	50	48	16
s1488	75	56	33	19
s3271	215	79	72	3
s3330	245	66	59	8
s3384	244	101	27	8
s4863	291	111	142	9
s5378	370	65	25	1
s9234	666	212	249	6
s15850	1171	478	442	4
s35932	2162	523	360	4

We can see from Table 3 that only a small fraction of latches needs to be replaced. This implies that if a flop is implemented as two back-to-back latches, the area overhead due to the replacement is reasonably small.

8 Conclusion

The trade-off 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 [10, 22]. 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 be easily extended to utilize other discrete coupling models, to include clock skew uncertainty, and can equally apply to crosstalk induced by capacitive and inductive couplings.

References

[1] R. Arunachalam, R. D. Blanton, and L. T. Pileggi. False coupling interactions in static timing analysis. In *DAC*, pages 726–731, 2001.

[2] R. Arunachalam, K. Rajagopal, and L. T. Pilleggi. Taco: Timing analysis with coupling. In *DAC*, pages 266–269, Los Angeles, CA, June 2000.

[3] P. Chen and K. Keutzer. Toward true crosstalk noise analysis. In *ICCAD*, pages 132–137, 1999.

[4] P. Chen, D. A. Kirkpatrick, and K. Keutzer. Switching window computation for static timing analysis in presence of crosstalk noise. In *ICCAD*, San Jose, CA, November 2000.

[5] P. Chen, Y. Kukimoto, C.-C. Teng, and K. Keutzer. On convergence of switching windows computation in presence of crosstalk noise. In *ISPD*, pages 84–89, 2002.

[6] F. Dartu and L. T. Pileggi. Calculating worst-case gate delays due to dominant capacitance coupling. In *DAC*, pages 46–51, Anaheim, CA, June 1997.

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

[8] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi. Determination of worst-case aggressor alighment for delay calculation. In *ICCAD*, pages 212–219, San Jose, CA, November 1998.

[9] S. Hassoun. Critical path analysis uing a dynamically bounded delay model. In *DAC*, pages 260–265, Los Angeles, CA, June 2000.

[10] S. Hassoun, C. Cromer, and E. C-Gamez. Static Timing analysis for level-clocked circuits in the presence of crosstalk. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 22(9):1270–1277, September 2003.

[11] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou. Optimizing two-phase, level-clocked circuitry. *JACM*, 44(1):148–199, 1997.

[12] 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 *ICCAD*, pages 552–555, November 1990.

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

[14] S. S. Sapatnekar and R. B. Deokar. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE TCAD*, 15(10):1237–1248, October 1996.

[15] Semiconductor Industry Association, http://public.itrs.net. *International Technology Roadmap for Semiconductors*, 2001.

[16] N. Shenoy and R. K. Brayton. Graph algorithms for clock schedule optimization. In *ICCAD*, 1992.

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

[18] S. Sirichotiyakul, D. Blaauw, C. Oh, R. Levy, V. Zolotov, and J. Zuo. Driver modeling and alignment for worst-case delay noise. In *DAC*, pages 720–725, 2001.

[19] T. G. Szymanski. Computing optimal clock schedules. In *DAC*, 1992.

[20] T. G. Szymanski and N. Shenoy. Verifying clock schedules. In *ICCAD*, 1992.

[21] T. Xiao and M. Marek-Sadowska. Functional correlation analysis in crosstalk induced critical paths identification. In *DAC*, pages 653–656, 2001.

[22] H. Zhou. Clock schedule verification with crosstalk. In *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 2002.

[23] H. Zhou. Timing analysis with crosstalk is a fixpoint on a complete lattice. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 22(9):1261–1269, September 2003.