Timing Analysis with Clock Skew

David Harris
Harvey Mudd College
301 E. Twelfth St.
Claremont, CA 91711
(909) 607-3623
David_Harris@hmc.edu

Mark Horowitz
Stanford University
Gates Room 306
Stanford, CA 94305
(650) 725-3707
horowitz@vlsi.stanford.edu

Dean Liu
Stanford University
Gates Room 320
Stanford, CA 94305
(650) 725-3657
dliu@stanford.edu

Clock skew is an increasing concern for high speed circuit designers. Circuit designers use transparent latches and skew-tolerant domino circuits to hide clock skew from the critical path and take advantage of clock domains to budget less skew between nearby elements than across the entire die. Unfortunately, current timing analysis algorithms do not handle clock skew. This paper extends the latch-based timing analysis of Sakallah, Mudge, and Olukotun to include different amounts of clock skew between different elements and presents an algorithm for verifying the constraints. It also offers a simple formulation of min-delay analysis including clock skew. With the less conservative skew budgets enabled by better timing analysis, we expect clocked systems will remain viable to multi-GHz frequencies.
Keywords: timing analysis, transparent latches, clock skew

1. Introduction

Clock skew is an increasing concern for high speed circuit designers. Cycle times have been dramatically shrinking, driven both by faster raw gate delays and by more aggressive designs using fewer gates per cycle [2]. Unfortunately, clock skew, the difference between actual and nominal interarrival times of a pair of clock signals, depends on die size, process and environmental variations, wire RC delay, and clock loading, all of which have been increasing. Therefore, designers have been forced to spend more power and area on the clock network and expect that clock skew as a fraction of cycle time will increase.

In systems built with normal flip-flops or traditional domino design techniques [15], clock skew directly reduces the amount of the cycle time available for useful computation. This is too much overhead for aggressive designs, so better circuit techniques which tolerate clock skew are gaining popularity. In systems built from transparent latches or skew-tolerant domino, reasonable amounts of clock skew have no impact on cycle time as long as data arrives when the latch is transparent or domino gate is evaluating. As clock frequencies reach the multi-GHz regime, skew from one corner of the die to another may still be difficult to tolerate. Fortunately, such fast systems can be divided into clock domains such that less clock skew must be budgeted between nearby gates than across the entire die [2].

Timing analysis addresses the question of whether a particular circuit will meet a timing specification. The analysis must check maximum delays to verify that a circuit will meet setup times at the desired frequency, and minimum delays to verify that hold times are satisfied. This paper describes how to extend a traditional formulation of timing analysis to handle clock skew, including different budgets for skew between different regions of a system.

We begin by reviewing previous work in Section 2, particularly the formulation of latch-based timing analysis from Sakallah *et al.* [9]. We build upon this formulation in Section 3 to analyze systems with clock skew. We can easily analyze systems with a single clock domain by adding worst case skew to the setup time of each latch. We then solve the more interesting case which allows different amounts of clock skew between different latches by introducing a vector of arrival times initiated by various clock edges. This leads to an explosion of timing constraints, but most are not tight. In Section 4, we present an extension of the Szymanski-Shenoy timing verification algorithm [11] which prunes the irrelevant constraints. Now that we have solved the problem of max-delay, we show a simple solution to min-delay analysis in Section 5. Section 6 evaluates the relaxation algorithm on a large example, showing that increase in complexity is small. Finally, Section 7 summarizes the work and concludes the paper.

2. Background

We begin by reviewing some of the key developments in timing analysis, then look more closely at the formulation presented by Sakallah, Mudge, and Olukotun, which handles level-sensitive latches. Section 3 will build upon this formulation to analyze systems with clock skew.

2.1 Previous Work

Early efforts in timing analysis, surveyed in [4], only considered edge-triggered flip-flops. Thus they only had to analyze the combinational logic blocks because the cycle time is set by the longest combinational path between registers. Netlist-level timing analyzers, such as CRYSTAL [8] and TV [6], used switch-level RC models to compute delay through the blocks.

Many circuits use level-sensitive latches instead of flip-flops. Latches complicate the analysis because they allow time borrowing: a signal which reaches the latch input while the latch is transparent does not have to wait for a clock edge, but rather can immediately propagate through the latch and be used in the next phase of logic. Analysis of systems with latches was long considered a difficult problem [8] and various netlist-level timing analyzers applied heuristics for latch timing, but eventually Unger

[14] developed a complete set of timing constraints for two-phase clocking with level-sensitive latches. LEADOUT [12], by Szymanski, checked timing equations to properly handle multiphase clocking and level-sensitive latches. Champernowne *et al.* [2] developed a set of latch to latch timing rules which allow a hierarchy of clock skews but did not permit time borrowing.

Sakallah, Mudge, and Olukotun [9] provide a very elegant formulation of the timing constraints for latch-based systems. They show that maximum delay constraints can be expressed with a system of inequalities. They then use a linear programming algorithm to minimize the cycle time and to determine an optimal clock schedule. Since the clock schedule is usually fixed and the user is interested in verifying that the circuits can operate at a target frequency, more efficient algorithms can be used to process the constraints, such as the relaxation approach suggested by Szymanski and Shenoy [11], [10]. Moreover, many of the constraints in the formulation may be redundant, so graph-based techniques proposed by Szymanski [13] can determine the relevant constraints. Ishii *et al* [5] offer yet another algorithm for verifying the cycle time of two-phase systems. Burks *et al.* [1] express timing analysis in terms of critical paths and support a limited model of clock skew.

2.2 Timing Analysis Formulation

The simplicity of the latch-based timing analysis formulation from Sakallah *et al.* [9] stems from a careful choice of time variables describing data inputs and outputs of the latches. In this paper, we consider only D-type latches with data in, data out, and clock terminals. It is easy to extend the model to other clocked elements such as flip-flops and domino gates.

A system contains a set of clocks $C=\{\phi_1, \phi_2, ..., \phi_k\}$ with a common cycle time and a set of latches $L=\{L_1, L_2, ..., L_l\}$. Without loss of generality, assume all clock phases are active high; *i.e.*, latches are transparent when the controlling phase is high. Define the following clock variables which are illustrated in Figure 1 for a two-phase system with 50% duty cycle clocks.

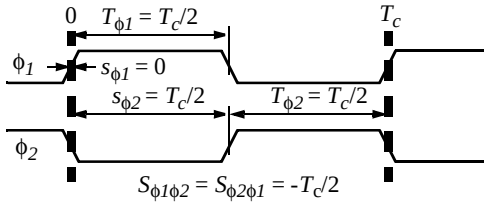

Figure 1. Two Phase Clock Waveforms

- T_c : clock cycle time, or period
- T_{ϕ_i} : duration for which ϕ_i is high
- s_{ϕ_i} : start time, relative to the beginning of the common clock cycle, of ϕ_i being high
- $S_{\phi_i \phi_j}$: a phase shift operator describing the difference in start time from ϕ_i to the next occurrence of ϕ_j. $S_{\phi_i \phi_j} \equiv s_{\phi_i} - (s_{\phi_j} + WT_c)$, where W counts cycle crossings between the clocks. Note that $S_{\phi_i \phi_i} = -T_c$ because it is the shift between subsequent rising edges of clock phase ϕ_i.

For each of the l latches in the system, define the following latch timing variables and parameters:

- p_i: clock phase used to control latch i
- A_i: arrival time, relative to the start time of p_i, of a valid data signal at the input to latch i
- D_i: departure time, relative to the start time of p_i, at which the signal available at the data input of latch i starts to propagate through the latch
- Q_i: output time, relative to the start time of p_i, at which the signal at the data output of latch i starts to propagate through the succeeding stages of combinational logic
- Δ_{DCi} : setup time for latch i required between the data input and the trailing edge of the clock input
- Δ_{DQi} : maximum propagation delay of latch i from the data input to the data output while the clock input is high

Finally, define the propagation delays between latch pairs:

- Δ_{ij} : maximum propagation delay through combinational logic between latch i and latch j. If there are no combinational paths from latch i to latch j, $\Delta_{ij} \equiv -\infty$ effectively eliminates the path from consideration.

Using these definitions, Sakallah *et al.* express constraints on the propagation of signals between latches and the setup of signals before the sampling edges of the latches.

Setup time constraints require that a signal arrive at a latch some setup time before the sampling clock edge. Thus:

$$\forall i \in L \qquad A_i + \Delta_{DCi} \le T_{p_i} \tag{1}$$

The propagation constraints relate the departure, output, and arrival times of latches. Data departs a latch input when the data arrives and the latch is transparent:

$$\forall i \in L \qquad D_i = \max(0, A_i) \tag{2}$$

The latch output becomes valid some latch propagation delay after data departs the input:

$$\forall i \in L \qquad Q_i = D_i + \Delta_{DQi} \tag{3}$$

Finally, the arrival time at a latch is the latest of the arrival times from data leaving other latches and propagating through combinational logic to the latch of interest. Notice that the phase shift operator S translates between relative times of the launching and receiving latch clocks.

$$\forall i, j \in L \qquad A_i = \max(Q_j + \Delta_{ji} + S_{p_j p_i}) \tag{4}$$

Observe that both D_i and Q_i will always be nonnegative quantities because a signal may not begin propagating through a latch until the clock has risen. A_i is unrestricted in sign because the input data may arrive before or after the latch clock. Assuming that clock pulse widths T_i are always greater than latch setup times Δ_{DCi} and eliminating Q and A, we can rewrite these constraints exclusively in terms of

signal departure times and the clock parameters:

L1. Setup Constraints:

$$\forall i \in L \qquad D_i + \Delta_{DCi} \le T_{p_i} \tag{5}$$

L2. Propagation Constraints:

$$\forall i, j \in L \qquad D_i = \max(0, \max(D_j + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i})) \tag{6}$$

Note that when there is no combinational path between latches *i* and *j*, $\Delta_{ij} \equiv -\infty$ and Equation 6 is trivially satisfied.

The minimum cycle time can be computed by solving an optimization problem of minimizing T_c subject to latch constraints L1 and L2. Often the designer only cares whether a system can operate at a specified frequency, rather than knowing the minimum possible cycle time. This simpler timing verification problem can be solved more efficiently with relaxation algorithms [11], [1].

3. Timing Analysis with Clock Skew

Sakallah's formulation discussed in the previous section does not account for clock skew. Since clock skews are becoming increasingly important, we now examine how to include skew in timing analysis. We first describe a simple modification to the setup constraints which account for a single clock skew budget across the chip. Unfortunately, this is very pessimistic because most clocked elements see much less than worst case skew. Therefore, we develop an exact analysis allowing for different skews between each pair of clocks. This leads to an explosion in the number of timing constraints. In the next section, we will see that most constraints are unnecessary and will present an algorithm to dynamically prune such constraints.

3.1 Clock Skew

We have defined clock variables describing the nominal timing relationships between various clocks. In a real circuit, the timing relationships may be slightly different due to clock skew, which includes both systematic and random or time-varying components. Circuit designers can manage systematic clock skews by appropriately partitioning logic between clocked elements. Timing analyzers also easily handle systematic skews by defining multiple clocks which include the predicted skew. However, random clock skew is a serious problem because a latch input must be ready by the earliest time a clocked element may sample, yet the latch output may not be valid until the latest time the clocked element activates. This uncertainty must be budgeted in each path.

To model clock skew, we use a large set of physical clock signals, *C*, even when there are only a small number of distinct logical clock phases. Conceptually, it is easy to envision a unique clock for each latch, but one can quickly group clocks that have very small skew relative to each other into one clock to reduce the number of clocks. For example, the system in Figure 2 shows a microprocessor core with the ALU and data cache in separate clock domains. It uses clocks $C=\{\phi_{1a}, \phi_{1b}, \phi_{2a}, \phi_{2b}\}$ where ϕ_{1a} and ϕ_{1b} are nominally identical but located in different parts of the chip and subject to skew. Only a small t_{skew}^{local} exists between clocks in the same domain, but the larger t_{skew}^{global} may occur between clocks in different domains.

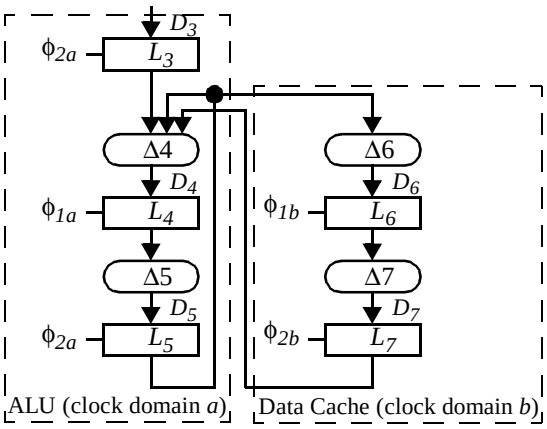

Figure 2. Example circuit with clock domains

For any two clocks ϕ_i and ϕ_j, the skew between particular edges of the two clocks is the absolute value of the difference between the nominal and actual interarrival times measured at any latches served by the clocks. For design purposes, it is most useful to know an upper bound on the skew between two clocks, $t_{skew}^{\phi_i, \phi_j}$, which is the maximum value of skew between any two edges of the clocks. Notice that skew is the positive difference between the two clock positions, rather than being plus or minus from a reference point. When using this information in a design, we assume the worst; *i.e.* for setup time checks, that the receiving clock is skewed early relative to the launching clock. This model of skew is more powerful than the min/max skew model of [1] because it accounts for correlations and lower skew between nearby clocks sharing parts of the distribution network. If skews are not symmetrical around the nominal interarrival times, we can define skew as a range rather than an absolute value.

3.2 Single Skew Formulation

The simplest and most conservative way to accommodate clock skew in timing analysis is to use a single upper bound on clock skew. Suppose that we assume a worst case amount of clock skew, t_{skew}^{global}, may exist between any two clocked elements on an integrated circuit. Such skew can be accommodated in the analysis by modifying the setup time constraint [11]. Data must setup before the falling edge of the clock, yet there may be skew between launching and receiving elements such that the data was launched off a late clock edge and is sampled on an early edge. Therefore, we must add clock skew to the effective setup time. Propagation constraints are unchanged.

L1S. Setup Constraints with Single Skew:

$$\forall i \in L \qquad D_i + \Delta_{DCi} + t_{skew}^{global} \le T_{p_i} \tag{7}$$

3.3 Exact Skew Formulation

In a real clock distribution system, clock skews between adjacent elements are typically much less than skews between widely separated elements. We can avoid budgeting global skew in all paths by considering the actual launching and receiving elements and only budgeting the possible skew which exists between the elements.

Unfortunately, the transparency of latches makes this a complex problem. Consider the setup time on a signal arriving at latch L_4 in Figure 2. How much skew must be budgeted in the setup time? The answer depends on the skew between the clock which originally launched the signal and ϕ_{1a}, the clock which is receiving the signal. For example, the signal might have been launched from L_7 on the rising edge of ϕ_{2b}, in which case global skew must be budgeted. On the other hand, the signal might have been launched from L_5 on the rising edge of ϕ_{2a}, then propagated through L_6 and L_7 while both latches were transparent. In such a case, only local skew must be budgeted because the launching and receiving clocks are in the same local domain despite the fact that the signal propagated through transparent elements in a different domain. We see that exact timing analysis with varying amounts of skew between elements must track not only the accumulated delay to each element, but also the clock of the launching element.

To track both accumulated delay and launching clock, we can define a vector of arrival and departure times at each latch, with one dimension per clock in the system. These times are still nominal, not including skew.

- A_i^c : arrival time at latch *i*, relative to the beginning of p_i, of a valid data signal launched by clock *c*
- D_i^c : departure time, relative to the beginning of p_i, at which the signal launched by clock *c* and available at the data input of latch *i* starts to propagate through the latch

The setup constraints must budget the skew t_{skew}^{c,p_i} between the launching clock *c* and the sampling clock p_i:

$$\forall i \in L, c \in C \qquad D_i^c + \Delta_{DCi} + t_{skew}^{c,p_i} \le T_{p_i} \tag{8}$$

The arrival time at latch *i* for a path launched by clock *c* depends on the propagation delay and departure times from other latches for signals also launched by clock *c*:

$$\forall i, j \in L, c \in C \qquad A_i^c = \max(D_j^c + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i}) \tag{9}$$

If a latch is transparent when its input arrives, data should depart the latch at the same time it arrives and with respect to the same launching clock. If a latch is opaque when its input arrives, the path from the launching clock will never constrain timing and a new path should be started departing at time 0, launched by the latch's clock. Because of skew between the launching and receiving clocks, the receiving latch may be transparent even if the input arrives at a slightly negative time. To model this effect, we allow departure times with respect to a clock other than that which controls the latch to be negative, equal to the arrival times. Departure times with respect to the latch's own clock are strictly nonnegative. To achieve this, we define an identity operator I_{ϕ_1,ϕ_2} on a pair of clocks ϕ_1 and ϕ_2 which is the minimum departure time for a signal launched by one clock and received by the other: 0 if $\phi_1 = \phi_2$ and negative infinity if the clocks are different.

These setup, nonnegativity, and propagation constraints are summarized below. Notice that the number of constraints is proportional to the number of distinct clocks in the system and is *k* times greater than the skewless formulation. Also, notice that the constraints are orthogonal; there is no mixing of constraints from different launching clocks.

L1E. Setup Constraints with Exact Skew:

$$\forall i \in L, c \in C \qquad D_i^c + \Delta_{DCi} + t_{skew}^{c,p_i} \le T_{p_i} \tag{10}$$

L2E. Propagation Constraints with Exact Skew:

$$\forall i, j \in L, c \in C \qquad D_i^c = \max(I_{c,p_i}, \max(D_j^c + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i})) \tag{11}$$

An example may help explain negative departure times. Consider a path launched from L_6 in Figure 2 on the rising edge of ϕ_{1b}: $D_6^{\phi_{1b}} = 0$. Let the cycle time T_c be 10 units, and $t_{skew}^{\phi_{1b},\phi_{2b}}$ be 1. Therefore, ϕ_{2b} may transition up to one unit of time earlier or later than nominal, relative to ϕ_{1b}, as shown in Figure 3. Also, suppose the latch propagation delay is 0, so $A_7^{\phi_{1b}} = \Delta 7 - 5$. If $\Delta 7$ is less than 4, the signal arrives at L_7 before the latch becomes transparent, even under worst case clock skew. If $\Delta 7$ is between 4 and 6 units, corresponding to $A_7^{\phi_{1b}}$ in the range of -1 to 1, the signal arrives at L_7 when the latch might be transparent, depending on the actual skew between ϕ_{1b} and ϕ_{2b}. If $\Delta 7$ is between 6 and 9 units, the signal arrives at L_7 when the latch is definitely transparent. Since the signal may depart the latch at the same time as it arrives when the latch is transparent, the departure time $D_7^{\phi_{1b}}$ may physically be as early as -1. We allow the departure time to be arbitrarily negative; if it is more negative than -1, it will always be less critical than the path departing L_7 on the rising edge of ϕ_{2b}. Departure times must be nonnegative with respect to the clock controlling the latch; for example, $D_7^{\phi_{2b}} \ge 0$.

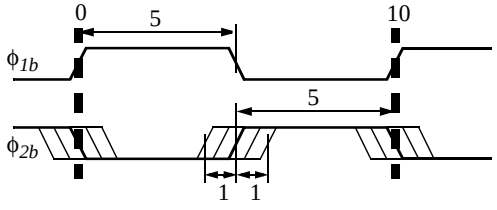

Figure 3. Clock waveforms including local skew

4. A Verification Algorithm

We extend the Szymanski-Shenoy timing verification procedure to handle arbitrary skews between elements and prune unnecessary constraints:

1 For each latch *i*:
2 $D_i^{p_i} = 0$; $D_i^{\max} = 0$; $c_i^{\max} = p_i$
3 Enqueue $D_i^{p_i}$
4 While queue is not empty
5 Dequeue D_j^c
6 For each latch *i* in fanout of *j*
7 $A = D_j^c + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_{i_{\max}}}$
8 If $(A > D_i^c)$ AND $\left(A + t_{\text{skew}}^{c_i^{\max}, c} > D_i^{\max}\right)$
9 If $\left(A + \Delta_{DCi} + t_{skew}^{c, p_i} > T_{p_i}\right)$
10 Report setup time violation
11 Else
12 $D_i^c = A$; Enqueue D_i^c
13 If $(A > D_i^{\max})$ $D_i^{\max} = A$; $c_i^{\max} = c$

This algorithm initializes the departure times from each latch with respect to its own clock to be zero. It also initializes a variable to track the latest departure time from the latch with respect to any clock. The algorithm then follows paths from each latch to its successors and computes the arrival time at the successors with respect to the launching clock. If this time is later than the latest departure time with respect to that clock which has been found so far at the receiving latch, the path may be important. However, if the arrival is earlier than the latest departure from the latch by more than the skew between the launching clock of the current path and the launching clock which determined the latest departure, this arrival will not cause any setup time violations unless the later departure also caused setup violations. Therefore, it can be ignored. This can be thought of as dynamically pruning constraints that will not be tight.

The algorithm performs a depth first path search if elements are enqueued at the head of the queue and a breadth first search if elements are enqueued at the tail. Breadth first is likely to be faster because it can prune paths earlier.

The algorithm is very similar to one which assumes no clock skew, but may take longer because it may search multiple paths through the same latch. This occurs when paths originating at different latches all arrive at a common latch at nearly the same time. Fortunately real systems tend to have a relatively small number of critical paths passing through any given latch so the runtime is likely to increase by much less than the number of constraints.

5. Min-Delay

Timing analyzers must not only compute long paths, but also short paths. Indeed, short paths are more serious because a chip can operate at reduced clock frequency if paths are longer than predicted, but will not operate at any frequency if min-delay constraints are not met. Such min-delay analysis checks that data launched from one latch or flip-flop will not propagate through logic so quickly as to violate the hold time of the next clocked element. Therefore, min-delay analysis only must check from one element to its successor; this is much easier than cycle time analysis in which a path may borrow time through many transparent latches.

To avoid min-delay failure, also known as "race-through" or "double-clocking," data departing one element must encounter enough delay that it does not violate the hold time of the next element. The earliest data could possibly depart an element is at time 0 with respect to the element's local clock; this earliest time is guaranteed to occur if the chip is run at reduced frequency where no time borrowing occurs. We define hold time and minimum propagation delays:

- Δ_{CDi} : hold time for latch *i* required between the falling edge of the clock and the time data changes again.
- δ_{DQi} : minimum propagation delay of latch *i* from the data input to the data output while the clock is high.
- δ_{ij} : minimum propagation delay through combinational logic between latch *i* and latch *j*. If there are no combinational paths from latch *i* to latch *j*, $\delta_{ij} \equiv \infty$.

Equation 12 describes this min-delay constraint between adjacent latches. A circuit is safe from race-through if, for every consecutive pair of latches, data from the earlier element cannot arrive at the later element until some hold time after the later element sampled. In the worst case, data departs one element at time zero and arrives at the next after the minimum propagation delay through the element and combinational logic adjusted by the phase shift operator to the receiver's clock. Data must not arrive at the receiver until a hold time after its sampling edge of the previous cycle; clock skew between the launching and receiving clocks effectively increases the hold time.

$$\forall i, j \in L \qquad \delta_{DQj} + \delta_{ji} + S_{p_j p_i} \geq T_{p_i} + \Delta_{CDi} + t_{skew}^{p_i, p_j} - T_c \quad (12)$$

Good estimates of the skew between launching and receiving clocks makes guaranteeing min-delay constraints easier than when worst case skew is assumed even between nearby elements. Min-delay can be checked by using standard algorithms such as those in [11] with an effective hold time equal to the sum of the actual hold time and skew.

6. Results

To evaluate the costs and benefits of the exact formulation, we analyzed a timing model of MAGIC, the Memory and General Interconnect Controller of the FLASH supercomputer [7], implemented in a 0.6μ process. After trimming false paths, we found 1819 latches and 10559 flip-flops connected by 593153 paths (Model A). To obtain an entirely latch-based design, we replaced each flip-flop with

a pair of latches and divided the path delay between the two latches, obtaining a system with 22937 latches (Model B). The chip was partitioned into ten units, each a local clock domain. We assumed 500 ps of global skew and 250 ps of local skew.

Table 1 shows the minimum cycle times achievable and number of latch departures enqueued in each run, a measure of the analysis cost. Model B is uniformly faster than Model A because latches allows the system to borrow time across cycles. The exact analysis shows that the system can run 50-90 ps faster than a single skew analysis conservatively predicts. Each departure is enqueued at least once when its departure time is initialized to 0. Paths borrowing time enqueue later departure times. The exact analysis also enqueues more latch departures because potentially critical paths from multiple launching clocks may pass through a single latch. The exact analysis enqueues 143 more than the single skew analysis in Model A and 333 more in Model B. These differences are only a few percent of the total number of departures, indicating that pruning makes the exact analysis only slightly more expensive than the single skew approximation. In all cases, the CPU time is under a second.

	Model A	**Model B**
Single Skew	9.43 ns 3866 departures	8.05 ns 24995 departures
Exact Skew	9.38 ns 4009 departures	7.96 ns 25328 departures

Table 1: Comparison of single and exact skew formulations

7. Conclusion

We believe that systems operating in the multi-GHz regime will be unable to achieve acceptably low global clock skews across the entire die. Instead of abandoning the synchronous paradigm for a fully asynchronous design, designers will divide the die into local clock domains offering smaller amounts of skew within each domain. Timing analyzers will need to recognize these domains and only budget the appropriate amount of clock skew.

We have extended the latch-based timing analysis formulation of Sakallah, Mudge, and Olukotun to handle clock skew, especially different amounts of clock skew between different elements. Allowing a single amount of clock skew everywhere effectively increases the setup time of each latch. An exact analysis allowing different amounts of skew between different elements involves tracking the clock which launched each path so that paths which leave a local skew domain and then return only budget the local skew. This leads to a multiplication of constraints proportional to the number of clocks. Fortunately, most constraints are not tight and can be dynamically pruned with an relaxation timing verification algorithm. Min-delay checks are simpler, effectively increasing the hold time by the clock skew. With the less conservative skew budgets enabled by better timing analysis, we expect clocked systems will remain viable to extremely high frequencies.

Acknowledgments

The authors wish to thank O. Olukotun, N. Shenoy, J. Avidan, R. McGowen, and M. Greenstreet for fruitful discussions about timing analysis.

8. Bibliography

[1] Burks, T., Sakallah, K., and Mudge, T., "Critical Paths in Circuits with Level-Sensitive Latches," *IEEE Trans. VLSI Sys*., vol. 3, no. 2, pp. 273-291, June 1995.

[2] Champernowne, A., Bushard, L., Rusterholtz, J., and Schomburg, J., "Latch-to-Latch Timing Rules," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 798-808, June 1990.

[3] Gronowski, P., et al., "A 433-MHz 64-b Quad-Issue RISC Microprocessor," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1687-1696, Nov. 1996.

[4] Harris, D., and Horowitz, M., "Skew-Tolerant Domino Circuits," *IEEE J. Solid-State Circuits*, vol. 32, no. 1, pp. 1702-1711, Nov. 1997.

[5] Ishii, A., Leiserson, C., and Papaefthymiou, M., "Optimizing Two-Phase, Level-Clocked Circuitry," in *J. ACM*, vol. 44, no. 1, pp. 148-199, Jan. 1997.

[6] Jouppi, N., *Timing Verification and Performance Improvement of MOS VLSI Designs*, Ph.D. thesis, Stanford University, 1984.

[7] Kuskin, J., et al., "The Stanford FLASH Multiprocessor," in *Proc. Intl. Symp. Comp. Arch*., pp. 302-313, Apr. 1994.

[8] Ousterhout, J., "A Switch-Level Timing Verifier for Digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 3, pp. 336-349, July 1985.

[9] Sakallah, K., Mudge, T., and Olukotun, O., "Analysis and Design of Latch-Controlled Synchronous Digital Circuits," *IEEE Trans. Computer-Aided Design*, vol. 11, no. 3, pp. 322-333, March 1992.

[10] Shenoy, N., Brayton, R., and Sangiovanni-Vincentelli, A., "A Pseudo-Polynomial Algorithm for Verification of Clocking Schemes," in *Tau 92*, 1992.

[11] Szymanski, T. and Shenoy, N., "Verifying clock schedules," in *ICCAD Tech. Papers*, pp. 124-131, Nov. 1992.

[12] Szymanski, T., "LEADOUT: A Static Timing Analyzer for MOS Circuits," in *ICCAD-86 Dig. Tech. Papers*, 1986, pp. 130-133.

[13] Szymanski, T., "Computing Optimal Clock Schedules," in *Proc. 29th Design Autom. Conf.*, pp. 399-404, 1992.

[14] Unger, S. and Tan, C., "Clocking Schemes for High-speed Digital Systems," *IEEE Trans. Comput.*, vol. C-35, pp. 880-895, Oct 1986.

[15] Weste, N., and Eshraghian, K., *Principles of CMOS VLSI Design*, Reading, MA: Addison-Wesley, 1993.