Timing Analysis Including Clock Skew

David Harris, Mark Horowitz, *Senior Member, IEEE*, and Dean Liu

***Abstract*—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 shared portions of the clock network to budget less skew between nearby elements than across the entire die, but current timing analysis algorithms do not handle correlated clock skews. This paper extends the Sakallah–Mudge–Olukotun (SMO) latch-based timing analysis to include different amounts of clock skew between different elements. The key change is that departure times from each latch must be defined with respect to launching clocks so that the skew between the launching and receiving clocks can be determined at each receiver. The exact analysis leads to an explosion in the number of timing constraints, but most constraints are not tight in practical situations and a modified version of the Szymanski–Shenoy relaxation algorithm gives exact results with only a small increase in runtime. The timing analysis formulation also captures the effects of skew on edge-triggered flip-flops, domino circuits, and min-delay constraints. Our exact algorithm, applied to a supercomputer node controller with over 12 000 clocked elements, finds the system can run 50–90 ps faster than a single skew analysis would predict and requires searching fewer than 4% more latch departures than conventional algorithms. With the less conservative skew budgets enabled by better timing analysis, we expect clocked systems will remain viable to multi-GHz frequencies.**

***Index Terms*—Clock skew, domino, min-delay, timing analysis, transparent latches.**

I. 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 [1]. Unfortunately, clock skew, the difference between actual and nominal interarrival times of a pair of clock signals, depends on process and environmental variations, wire RC delay, and clock loading, all of which have been increasing relative to gate delays. 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 [2], clock skew directly reduces the amount of the cycle 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, many paths involve clocked elements sharing a common local clock buffer and, therefore, see less skew. We can take advantage of this common case to obtain less-conservative skew budgets.

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 extends 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 II, particularly the formulation of latch-based timing analysis from Sakallah *et al.* [4]. We build upon this formulation in Section III 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 develop an exact set of constraints for analyzing systems with different amounts of skew between different elements. This exact analysis leads to an explosion of the number of timing constraints. By introducing a hierarchy of clock domains with tighter bounds on skews within smaller domains, we offer an approximate analysis with fewer timing constraints which is conservative, but less pessimistic than the single skew scenario. In Section IV, we apply these various formulations to analyze an abstract processor core. Many real systems include not only transparent latches, but also edge-triggered registers and domino gates. Therefore, we extend the formulation to handle such clocked elements in Section V. Min-delay analysis, described in Section VI, is simpler because it involves only one constraint between pairs of clocked elements even with skew. Section VII extends the Szymanski–Shenoy relaxation timing analysis algorithm to accommodate different amounts of skew between different elements. Although one could construct pathological cases which increase runtime proportional to the number of clocks in the system, most constraints are loose in real systems. Section VIII presents the results of timing analysis on a supercomputer node controller with over 12 000 clocked elements. The exact analysis shows cycle times up to 90-ps better than a single skew analysis would predict, while searching only 4% more paths. Finally, Section IX summarizes the work and concludes the paper.

Manuscript received December 8, 1998; revised January 12, 1999 This paper was recommended by Associate Editor T. Szymanski.

D. Harris is with Harvey Mudd College, Claremont, CA 91711 USA (e-mail: david_harris@hmc.edu).

M. Horowitz and D. Liu are with the Center for Integrated Systems, Stanford University, Stanford, CA 94305 USA.

Publisher Item Identifier S 0278-0070(99)09477-4.

II. Background

We begin by reviewing some of the key developments in timing analysis, then look more closely at the formulation presented by Sakallah *et al.* [4], which handles level-sensitive latches. In the next section, we will build upon this formulation to analyze systems with clock skew.

A. Previous Work

Early efforts in timing analysis, surveyed in [6], only considered edge-triggered flip-flops. Thus, they had to analyze just the combinational logic blocks between registers because the cycle time is set by the longest combinational path between registers. Netlist-level timing analyzers, such as CRYSTAL [7] and TV [8], used switch-level RC models [9] to compute delay through the combinational 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 [7] and various netlist-level timing analyzers applied heuristics for latch timing, but eventually Unger [10] developed a complete set of timing constraints for two-phase clocking with level-sensitive latches. LEADOUT [11], by Szymanski, checked timing equations to properly handle multiphase clocking and level-sensitive latches. Champernowne *et al.* [3] developed a set of latch to latch timing rules which allow a hierarchy of clock skews but did not permit time borrowing.

Sakallah *et al.* [4] 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 [5]. Moreover, many of the constraints in the formulation may be redundant, so graph-based techniques proposed by Szymanski [12] can determine the relevant constraints. Ishii *et al.* [13] offer yet another efficient algorithm for verifying the cycle time of two-phase latched systems. Burks *et al.* [14] express timing analysis in terms of critical paths and support a min/max model of clock skew, though the model does not reflect correlations between clocks which reduce local skew.

B. Timing Analysis Formulation

The simplicity of the latch-based timing analysis formulation from Sakallah *et al.* [4] stems from a careful choice of time variables describing data inputs and outputs of the latches. In this section, we consider only D-type latches with data in, data out, and clock terminals. Section V extends the model to include other clocked elements such as flip-flops and domino gates.

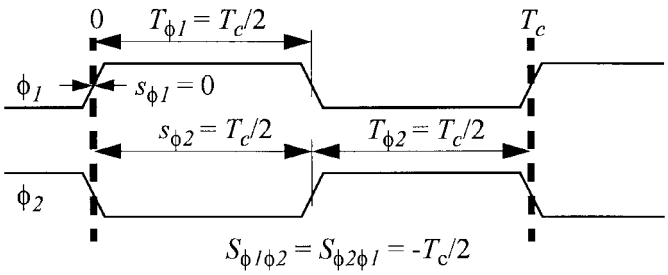

Fig. 1. Two-phase clock waveforms.

A system contains a set of clocks $C = \{\phi_1, \phi_2, \cdots, \phi_k\}$ with a common cycle time and a set of latches $L = \{L_1, L_2, \cdots, L_l\}$. 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 describing the cycle time and the nominal waveforms of each clock. These variables are illustrated in Fig. 1 for a two-phase system with 50% duty cycle clocks.

- 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.
- $S_{\phi_i \phi_j}$ 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 clocks. Note that $S_{\phi_i \phi_i} = -T_c$ because it is the shift between consecutive rising edges of clock phase ϕ_i.

For each of the l latches in the system, define the following variables and parameters that describe which clock is used to control each latch, when data arrives and departs each latch, and the setup time and propagation delay of each latch.

- 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 pairs of latches

- 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 by ensuring it will never be critical.

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. These constraints are given as equalities involving the max function.

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

$$\forall i \in L \quad A_i + \Delta_{DCi} \leq T_{p_i}. \tag{1}$$

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

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

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

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

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

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

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 the Q and A variables, we can rewrite these constraints exclusively in terms of signal departure times and the clock parameters.

L1. Setup Constraints:

$$\forall i \in L \quad D_i + \Delta_{DCi} \leq T_{p_i}. \tag{5}$$

L2. Propagation Constraints:

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

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 is only interested in 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 [5], [14].

III. Timing Analysis with Clock Skew

The 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 review 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. Next 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. By making a simple approximation of clock domains, we finally formulate the problem in a way which is conservative, yet less pessimistic than the single skew approach.

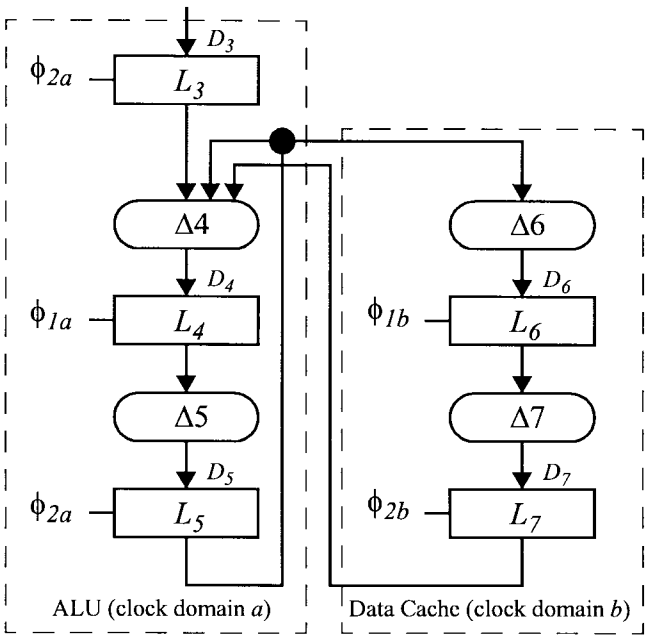

Fig. 2. Example circuit with clock domains.

A. 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 and may even intentionally introduce skew to provide more time in a cycle for a critical path. 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.

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 Fig. 2 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. Systematic skews are accounted for by defining multiple physical clocks so only unpredictable components of skew appear in the skew budgets.

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: for maximum delay (setup time) checks, that the receiving clock is skewed early relative to the launching clock; and for minimum delay (hold time) checks, that the receiving clock is skewed late relative to the launching clock. This model is more powerful than the min/max skew model of Burks *et al.* [14] because it supports correlations and lower skew between nearby clocks sharing part 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.

B. 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 [5]. 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. The propagation constraints are unchanged because the logic delay between latches is independent of skew.

L1S. Setup Constraints with Single Skew:

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

C. Exact Skew Formulation

In a real clock distribution system, clock skews between adjacent elements sharing a local clock generator 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 Fig. 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, relative to the beginning of p_i, of a valid data signal launched by clock c and now at the input to latch i.
- 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 element controlled by p_i

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

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

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

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 zero, 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: zero if $\phi_1 = \phi_2$ and negative infinity if the clocks are different.

These setup 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 Analysis:

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

L2E. Propagation Constraints with Exact Skew Analysis:

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

An example may help explain negative departure times. Consider a path launched from L_6 in Fig. 2 on the rising edge of ϕ_{1b}: $D_6^{\phi_{1b}} = 0$. Let the cycle time T_c be ten units, and $t_{skew}^{\phi_{1b},\phi_{2b}}$ be one. Therefore, ϕ_{2b} may transition up to one unit of time earlier or later than nominal, relative to ϕ_{1b}, as shown in Fig. 3. Also, suppose the latch propagation delay is zero, so $A_7^{\phi_{1b}} = \Delta 7 - 5$. If $\Delta 7$ is less than four, the signal arrives at L_7 before the latch becomes transparent, even under worst case

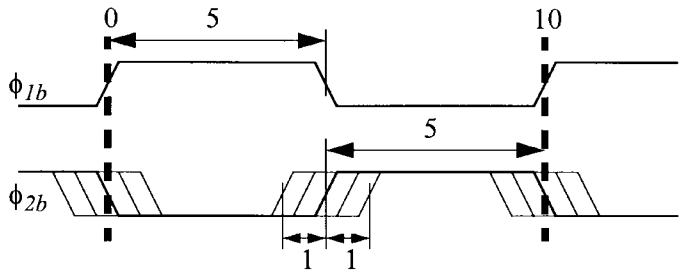

Fig. 3. Clock waveforms including local skew.

clock skew. If $\Delta 7$ is between four and six 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 six and nine 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 be physically 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} because it will arrive before L_7 becomes transparent. Departure times must be nonnegative with respect to the clock controlling the latch; for example, $D_7^{\phi_{2b}} \geq 0$.

D. Clock Domain Formulation

The exact timing analysis formulation leads to an explosion in the number of constraints required for a system with many clocks; a system with k clocks has k times as many constraints as the single skew formulation. We would like to develop an approximate analysis which gives more accurate results than the single skew formulation, yet has fewer constraints than the exact formulation. To do this, we will use the concepts of skew hierarchies and clock domains [3].

We wish to take advantage of the fact that the skew between nearby clocks is smaller than the skew between clocks in opposite corners of the die without introducing a large number of constraints by tracking the precise launching clock of every path. We, therefore, define a skew hierarchy, which is a collection of sets of clocks in the system. The sets are called clock domains. Each clock domain $d \subset C$ of the hierarchy has an associated number h which is called the level of the clock domain. A skew hierarchy has n levels, where level-1 clock domains are the smallest domains and the level n domain contains all the clocks C of the system. Define $H = \{1, \cdots, n\}$ to be the set of levels. Clock domains do not partially overlap; in other words, for any pair of clock domains, either one is a subset of the other or the domains are disjoint. If one domain contains another, the larger domain has the higher level. $n = 1$ corresponds to the single skew approximation. $n = 2$ is another interesting case, corresponding to a system with local and global skews. We define the following skew hierarchy variables.

t_{skew}^h Upper bound on skew between two clocks in a level h clock domain. This quantity monotonically increases with h. The top level domain experiences global skew: $t_{skew}^n = t_{skew}^{global}$.

h_{ij} Level of the smallest clock domain containing clocks i and j, i.e., the minimum h s.t. $t_{skew}^h \geq t_{skew}^{i,j}$.

Skew hierarchies apply especially well to systems constructed in a hierarchical fashion. For example, Fig. 4 illustrates an H-tree clock distribution network. It attempts to provide a two-phase clock consisting of ϕ_1 and ϕ_2 to the entire chip with zero skew. Although there are only two phases, the system actually contains 16 clocks for the purpose of modeling skew. All of the wire lengths in principle can be perfectly matched, so it is ideally possible to achieve zero systematic clock skew in the global distribution network. Even so, there is some RC delay along the final clock wires. Also, process and environmental variation in the delays of wires and buffers in the distribution network cause random clock skew. The clock skews between various phases depend on the level of their common node in the H-tree. For example, ϕ_{1ltl} and ϕ_{2ltl} only see a small amount of skew, caused by the final stage buffers and local routing. On the other hand, ϕ_{1ltl} and ϕ_{1rbr} on opposite corners of the chip may experience much more skew. The boxes show how the clocks could be collected into a five-level skew hierarchy.

The concept of skew hierarchies also applies to other distribution systems. For example, in a grid-based clock system, as used on the DEC Alpha 21164 [15], local skew is defined to be the RC skew between elements in a 500-μm radius, while global skew is defined to be the RC skew between any clocked elements on the die. Global skew is 90 ps, while local skew is only 25 ps. Therefore, the chip could be partitioned into distinct 500-μm blocks so that elements communicating within blocks only see local skew, while elements communicating across blocks experience global skew.

Now that we have defined skew hierarchies and clock domains, we return to the timing analysis approximation. The problem of an excessive number of timing constraints occurs because we must track the launching clock of each path so that when the path crosses to another clock domain, then returns to the original domain, only local skew must be budgeted at the latches in the original domain. An alternative is to only track whether a signal is still in the same domain as the launching clock or if it has ever crossed out of the local domain. In the first case, we budget only local clock skew. In the second case, we always budget global clock skew, even if the path returns to the original domain. This is conservative; for example, in Fig. 2, a path which starts in the arithmetic logic unit (ALU), then passes through the data cache while the cache latches are transparent and returns to the ALU would unnecessarily budget global skew upon return to the ALU. However, it greatly reduces the number of constraints, since we must only track whether the path should budget global or local skew, leading to only twice as many constraints as the single skew formulation. In general, we can extend this approach to handle n levels of hierarchical clock domains.

Again, we define multiple departure times, now referenced to the clock domain level of the signal rather than to the launching clock.

A_i^h Arrival time, relative to the beginning of p_i, of a valid data signal on a path which has crossed clock domains

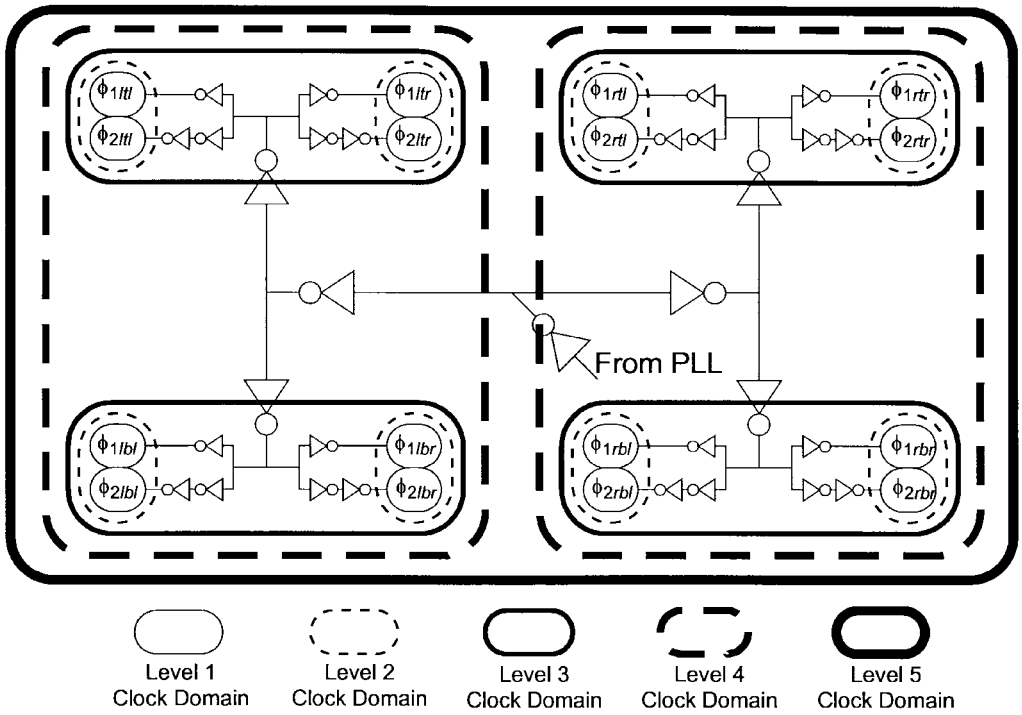

Fig. 4. H-tree clock distribution network.

at level h of the clock domain hierarchy and is now at the input to latch i.

D_i^h Departure time, relative to the beginning of p_i, at which the signal which has crossed clock domains at level h of the clock domain hierarchy and is now available at the data input of latch i starts to propagate through the latch.

When a path crosses clock domains, it is bumped up to budget the greater skew; in other words, the skew level at the receiver is the maximum of the skew level of the launched signal and the actual skew level between the clocks of the departing and receiving latches. As usual, departure times with respect to the latch's own clock are strictly nonnegative while departure times with respect to other clocks may be negative. Because we do not track the actual launching clock, but treat all clocks within a level-1 clock domain the same, we require that departure times from level-1 domains be nonnegative. To achieve this, we define an identity operator I_h on a level of the skew hierarchy which is the minimum departure time for a departure time at that level of the hierarchy: zero for departures with respect to level-1, and negative infinity for departures with respect to higher levels.

The setup and propagation constraints are listed below. Notice that the number of constraints is now proportional only to the number of levels of the clock domain hierarchy, not the number of clocks or even the number of domains. For a system with two levels of clock domains, i.e., local and global, this requires only twice as many constraints as the single skew formulation.

L1D. Setup Constraints with Clock Domain Analysis:

$$\forall i \in L, h \in H \quad D_i^h + \Delta_{DCi} + t_{skew}^h \leq T_{p_i}. \tag{12}$$

L2D. Propagation Constraints with Clock Domain Analysis:

$$\forall i, j \in L, h_1 \in H, h_2 = \max(h_1, h_{p_i p_j})$$
$$D_i^{j_2} = \max(I_{h_2}, \max\left(D_j^{h_1} + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i}\right)) \tag{13}$$

Yet another option is to lump clocks into a modest number of local clock domains, then perform an exact analysis on paths which cross clock domains. The number of constraints in such an analysis is proportional to the number of local clock domains, which is smaller than the number of physical clocks required for exact analysis, but larger than the number of levels of clock domains. Paths within a local domain always budget local skew. This hybrid approach avoids unnecessarily budgeting global skew for paths which leave a local domain but return a receiver in the local domain.

IV. Example

Let us return to the microprocessor example of Fig. 2 to illustrate applying timing analysis to systems with four clocks and a two-level skew hierarchy. We will enumerate the timing constraints for each formulation, then solve them with a linear programming tool to obtain minimum cycle time. The example will illustrate time borrowing, the impact of global and local skews, and the conservative approximation made by the inexact algorithms.

Suppose the nominal clock and latch parameters are identical to those in the example of Section II, but that the system experiences $t_{skew}^{local} = 1$ of skew between clocks in a particular domain and $t_{skew}^{global} = 3$ of skew between clocks in different domains.

The timing constraints for each formulation were entered into a linear programming system. Various values of $\Delta 4$ to $\Delta 7$ were selected to test the analysis. The values were all selected

TABLE I
EXAMPLES OF TIMING ANALYSIS RESULTS

Δ4	Δ5	Δ6	Δ7	T_c exact	T_c clock domains	T_c single skew	Notes
5	5	5	5	10	10	10	balanced logic
6	3	6	5	10	10	10	time borrowing
0.5	9.5	2.5	5	10.5	10.5	**12.5**	local skew limit
2	8	5	5	10.67	10.67	**11**	global skew limit
8	2	5	5	11	11	11	global skew limit
7	2	6	5	10	**10.5**	**10.5**	conservative result

so that a cycle time of ten units could be achieved in the case of no skew. The examples illustrate well-balanced logic, time borrowing between phases and across cycles, cycles limited by local and global skews, and a case in which the clock domain analysis yields conservative results.

Table I shows the values of combinational logic delay and cycle times achieved in each example. Bold data indicates conservative results caused by inexact modeling. The clock domains results match the exact results in all cases but one, in which a path started in the ALU, passed through the cache while the latches were transparent, and returned to the ALU. Only local skew must be budgeted on return, but the clock domain analysis method conservatively budgeted global skew, leading to a pessimistic cycle time. The single skew formulation is conservative in three cases which used large amounts of time borrowing where only local skew actually applied but global skew was unnecessarily budgeted.

V. EXTENSION TO FLIP-FLOPS AND DOMINO CIRCUITS

So far, we have addressed the question of timing analysis for transparent latches. Pulsed latches have identical cycle time constraints as transparent latches and, therefore, are also handled. Edge-triggered flip-flops are even simpler because they do not allow time borrowing; therefore, the departure time from a flip-flop is always with respect to the flip-flop's own clock. We can also extend the framework to handle domino circuits, which may have the timing requirements of transparent latches or edge-triggered flip-flops, depending on the monotonicity of the inputs. The main change introduced in this section is to track both arrival and departure times, because inputs to edge-triggered devices must arrive some setup time before the edge and do not depart until after the edge. We present only the exact analysis; the formulation assuming clock domains is very similar.

A. Flip-flops

For flip-flops, data must arrive before the rising edge of the clock phase, rather than the falling edge. Let $F = \{F_1, F_2, \cdots, F_f\}$ be the set of flip-flops. Data always departs the flop at the rising edge. We must, therefore, separately track arrival and departure times and introduce a set of departure constraints which relate arrival and departure times. The setup and departure constraints are written differently for flip-flops and latches.

Setup Constraints for Flip-Flops:

$$\forall i \in F, c \in C \quad A_i^c + \Delta_{DCi} + t_{skew}^{p_i,c} \leq 0 \tag{14}$$

Setup Constraints for Latches:

$$\forall i \in L, c \in C \quad A_i^c + \Delta_{DCi} + t_{skew}^{p_i,c} \leq T_{p_i}. \tag{15}$$

Departure Constraints for Flip-Flops:

$$\forall i \in F \quad D_i^{p_i} = 0. \tag{16}$$

Note that there is no departure constraint from clocks other than the flop's launching clock because flip-flops are not transparent.

Departure Constraints for Latches:

$$\forall i \in L, c \in C \quad D_i^c = \max(I_{c,p_i}, A_i^c) \tag{17}$$

These departure constraints capture the nonnegativity constraint of the latch.

Propagation Constraints for All Elements:

$$\forall i \in L \cup F, j \in L \cup F, c \in C$$
$$A_i^c \geq D_j^c + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i} \tag{18}$$

Although this formulation has more variables than the formulations including only latches, it actually involves less computation because the arrival times of latches are just intermediate variables requiring no more computation and because flip-flop analysis is simpler than latch analysis since time borrowing never occurs. Also note that we can use the same setup and departure constraints for flip-flops as for latches if we substitute $T_i = 0$ for flip-flop clocks.

B. Domino Gates

Domino gates can easily be extended to this framework. When inputs to a domino gate are monotonically rising, they may arrive after the gate has entered evaluation and the domino gate may be modeled exactly as a latch. When the inputs to the domino gate are nonmonotonic, they must arrive before the gate has entered evaluation and the gate may be modeled as a flip-flop for cycle-time calculations, with the additional caveat that the inputs must not change while the gate is evaluating;

i.e., the hold time is quite long. Hold times only appear in min-delay calculations and are discussed in the next section. Additional constraints are added to ensure precharge finishes in time, as described in Venkat *et al.* [16] and Van Campenhout, *et al.* [17]. In summary, one can label each domino gate input as either monotonic or nonmonotonic model it as a latch or flip-flop input, accordingly, with additional constraints to ensure precharge is fast enough.

VI. 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 that data could possibly depart an element is at time zero 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 minimum propagation delays through the clocked element and combinational logic.

- Δ_{CDi} Hold time for latch i required between the falling edge of the clock input 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 input 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 (19) describes this min-delay constraint between adjacent latches and flip-flops. A circuit is safe from race-through if, for every consecutive pair of clocked elements, data from the earlier element cannot arrive at the later element until some hold time after the previous sampling edge of the later element. 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. Time is adjusted by the phase shift operator to be relative 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. As in Section V, we substitute $T_i = 0$ for edge-triggered flip-flops

$$\forall i, j \in L \cup F$$
$$\delta_{DQi} + \delta_{ji} + S_{p_j p_i} \geq T_i + \Delta_{CDi} + t_{skew}^{p_i, p_j} - T_c. \quad (19)$$

Note that the number of constraints does not grow with the number of clocks, so min-delay checks are fundamentally easier than setup time checks. Better estimates of the skew between launching and receiving clocks makes guaranteeing min-delay constraints easier. A conservative design may assume a single worst case skew between all elements on the chip; this leads to excessive minimum propagation delay requirements between elements. By computing actual skews between each clock, smaller skews can be budgeted between nearby elements.

```
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   For each flip-flop i:
5     D_i^{p_i} = 0
6     Enqueue D_i^{p_i}
7   While queue is not empty
8     Dequeue D_j^c
9     For each latch i in fanout of j
10      A = D_j^c + Δ_DQj + Δ_ji + S_{p_j p_{i_max}}
11      If (A > D_i^c) AND (A + t_diff^{c_i, c} > D_i^max)
12        If (A + Δ_DCi + t_skew^{c, p_i} > T_{p_i})
13          Report setup time violation
14        Else
15          D_i^c = A ; Enqueue D_i^c
16          If (A > D_i^max)
17            D_i^max = A ;  c_i^max = c
18    For each flip-flop i in fanout of j
19      A = D_j^c + Δ_DQj + Δ_ji + S_{p_j p_i}
20      If (A + Δ_DCi + t_skew^{c, p_i} > 0)
21        Report setup time violation
```

Fig. 5. Timing verification algorithm.

VII. A Verification Algorithm

Szymanski and Shenoy present a relaxation algorithm for verifying that timing constraints are met at a given cycle time assuming no clock skew [5]. We extend the algorithm to handle arbitrary skews between elements and prune unnecessary constraints, as shown in the pseudocode of Fig. 5. A key aspect of the algorithm is the introduction of extra variables for each latch, $D_i^{\max}$ and $c_i^{\max}$, which track the latest departure from a latch with respect to any launching clock so that other paths through the latch can be pruned if they cannot be critical.

Let us first see how this algorithm handles latches, then return to the simpler case of flip-flops. The algorithm initializes the departure times from each latch with respect to its own clock to be zero. It also initializes a variable $D_i^{\max}$ to track the latest departure time from the latch with respect to any

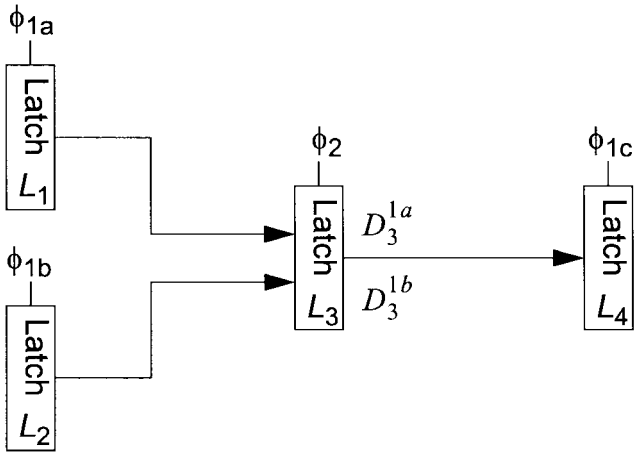

Fig. 6. Pruning of paths with different clock skews.

clock and a variable to track the clock that launched that latest departure (Step 2). 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 (Step 10).

A key idea of the algorithm is to prune paths which arrive early enough that they could not possibly be more critical than existing paths. To be potentially more critical and hence avoid pruning, an arrival time must satisfy two conditions (Step 11). One is that the arrival time must be later than all other departure times with respect to the same clock. The other is that the arrival time must potentially be as critical as the latest previously discovered departure time. If there were no clock skew or a single global skew budget everywhere, an arrival would only be more critical than the latest existing departure time if it actually were later: $A > D_i^{\max}$. However, we allow different amounts of skew between different clocks. Fig. 6 shows how this complicates our pruning:

Suppose that $t_{skew}^{\phi_{1a},\phi_{1c}} = 3$, while $t_{skew}^{\phi_{1b},\phi_{1c}} = 1$. Suppose that the departure time from L_3 D_3^{1b} on a path launched from L_2 is two units and that we find a path arrives at L_3 from L_1 at time one unit. Can we prune this path? If the clock skews were the same, we could because the path from L_1 arrives earlier than the path from L_2. Because the clock skews are different, however, data launched from L_1 must arrive at L_4 earlier than data launched from L_2. Therefore, the path from L_1 may also be critical even though its departure time is earlier.

Clearly, if one path arrives at a latch more than the worst case global clock skew before another, the early path cannot possibly be critical and may be trimmed. We can prune more aggressively by computing the difference in clock skews between a pair of launching clocks ϕ_i and ϕ_j and any possible receiving clock, $t_{diff}^{\phi_i,\phi_j}$

$$t_{diff}^{\phi_i,\phi_j} = \max(t_{skew}^{\phi_i,\phi_r} - t_{skew}^{\phi_j,\phi_r}) \quad \forall \phi_r \in C. \tag{20}$$

From this definition, we can show that $t_{diff}^{\phi_i,\phi_j} \leq t_{skew}^{\phi_i,\phi_j}$. Moreover, in a system budgeting a single global skew between all clocks, t_{diff} is zero and negative departure times never occur, agreeing with the single skew formulation.

In general, Step 11 checks this criteria, pruning all paths with arrival times before the latest departure by more than the clock skew between the launching clock c of the path under consideration and the launching clock $c_i^{\max}$ of the path causing the latest departure time. If the path is not pruned, it is checked for setup time violations and added to the queue so that paths to subsequent latches can be checked. Also, if it is later than the latest previously discovered departure time, it replaces the previous time (Step 17).

Flip-flops are handled in a similar fashion, but only have a departure time of zero with respect to their own clock because no time borrowing takes place. As discussed in Section V, domino gates are analyzed either as latches or flip-flops, depending on the monotonicity of the inputs.

The algorithm is very similar to one which assumes no clock skew, but may take longer because it may trace multiple paths through the same latch. This occurs when paths originating at different latches with skew between them all arrive at a common latch at nearly the same time. Fortunately, as we shall see, 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. Setup time constraints are only checked as paths are enqueued, so constraints involving pruned paths never need to be checked. Timing analysis with clock domains is similar to analysis with exact skew. The runtime may be somewhat improved because a hierarchy of h levels of clock domains must trace at most h paths through any given latch. Of course, the results are more conservative.

So far, we have addressed the question of verifying that a design meets a cycle time goal because this is the primary question asked by designers. It is also straightforward to compute the minimum cycle time of a design using Sakallah's linear programming approach [4]. The constraints as presented are not quite linear because they involve the max function. The max function can be replaced by multiple inequalities, transforming the constraints into linear inequalities while preserving the minimum cycle time. These inequalities can be solved by linear programming techniques. The clock domain formulation may be particularly relevant to linear programming because the number of constraints impacts runtime.

VIII. Results

To evaluate the costs and benefits of the exact formulation, we analyzed a timing model of the memory and general interconnect controller (MAGIC) of the FLASH supercomputer [18], implemented in a 0.6-μ CMOS process. MAGIC includes a two-way superscalar RISC processing engine and several large data buffers. We extracted a timing model from the standard delay format (SDF) data produced by LSI Logic tools, then trimmed long paths such as those involving reset or scan. After trimming, we found 1819 latches and 10 559 flip-flops connected by 593 153 combinational 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 22 937 latches (Model B). The chip was partitioned into ten units, each a local clock domain. We assumed 500 ps of global skew between domains and 250 ps of local skew within domains.

TABLE II
COMPARISON OF SINGLE AND EXACT SKEW FORMULATIONS

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

We applied the timing analysis algorithm of Section VII to the timing model. Table II 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 allow the system to borrow time across cycles, solving some critical paths. The exact analysis shows that the system can run 50–90 ps faster than a single skew analysis conservatively predicts. Each latch departure is enqueued at least once when its departure time is initialized to zero. 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 less than 4% 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 for analysis is under a second, much shorter than the time required to read the timing model from disk.

These results indicate that the exact skew formulation works well in practice because only a small fraction of paths require time borrowing and because an even smaller fraction of paths involve negative departure times. In this particular problem, no critical paths depart a clock domain and return to it, so the clock domain formulation would have found equally good cycle times. However, the cost of the exact skew formulation is low enough that no approximations are necessary. In an aggressive system such as a full-custom microprocessor in which a large number of paths narrowly meet timing, the benefits of budgeting exact skews are likely to be more significant because they allow the designer to spend less effort optimizing the paths within local clock domains.

IX. CONCLUSION

We expect 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 *et al.* to handle clock skew, particularly 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. Most practical systems use clocked elements besides just transparent or pulsed latches, so we also incorporate edge-triggered flip-flops and domino gates into the timing analysis formulation by separately tracking arrival and departure times at each clocked element. In addition to verifying cycle time, we check for min-delay violations, effectively increasing the hold time of each element by the potential clock skew between launching and receiving elements. We modified the Szymanski–Shenoy timing analysis algorithm to support different amounts of clock skew between different clocks. Although the runtime potentially increases with the number of clocks, we showed that a real design involved few negative departure times and thus required fewer than 4% more iterations for an exact analysis. With the less conservative skew budgets enabled by better timing analysis, we expect clocked systems will remain viable to extremely high operating frequencies.

ACKNOWLEDGMENT

The authors would like to thank O. Olukotun, N. Shenoy, J. Avidan, R. McGowen, and M. Greenstreet for fruitful discussions about timing analysis. The authors also thank the editor and anonymous reviewers for many helpful suggestions.

REFERENCES

[1] P. Gronowski, W. Bowhill, R. Preston, M. Gowan, and R. Allmon, "High-performance microprocessor design," *IEEE J. Solid-State Circuits*, vol. 33, pp. 676–686, May 1998.
[2] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*. Reading, MA: Addison-Wesley, 1993, p. 351.
[3] A. Champernowne, L. Bushard, J. Rusterholtz, and J. Schomburg, "Latch-to-latch timing rules," *IEEE Trans. Comput.*, vol. 39, pp. 798–808, June 1990.
[4] K. Sakallah, T. Mudge, and O. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 322–333, Mar. 1992.
[5] T. Szymanski and N. Shenoy, "Verifying clock schedules," in *ICCAD Dig. Tech. Papers*, Nov. 1992, pp. 124–131.
[6] R. B. Hitchcock, "Timing verification and timing analysis program," in *25 Years of Electronic Design Automation.* Piscataway, NJ: IEEE Press/ACM, 1988.
[7] J. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 336–349, July 1985.
[8] N. Jouppi, "Timing verification and performance improvement of MOS VLSI designs," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1984.
[9] J. Rubinstein, P. Penfield, and M. Horowitz, "Signal delay in RC tree networks," in *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.
[10] S. Unger and C. Tan, "Clocking schemes for high-speed digital systems," *IEEE Trans. Comput.*, vol. C-35, pp. 880–895, Oct 1986.
[11] T. Szymanski, "LEADOUT: A static timing analyzer for MOS circuits," in *ICCAD-86 Dig. Tech. Papers*, 1986, pp. 130–133.
[12] ——, "Computing optimal clock schedules," in *Proc. 29th Design Automation Conf.*,1992, pp. 399–404.
[13] A. Ishii, C. Leiserson, and M. Papaefthymiou, "Optimizing two-phase, level-clocked circuitry," in *J. ACM*, vol. 44, no. 1, pp. 148–199, Jan. 1997.
[14] T. Burks, K. Sakallah, and T. Mudge, "Critical paths in circuits with level-sensitive latches," *IEEE Trans. VLSI Syst.*, vol. 3, no. 2, pp. 273–291, June 1995.

[15] P. Gronowski *et al.*, "A 433-MHz 64-b quad-issue RISC microprocessor," *IEEE J. Solid-State Circuits*, vol. 31, pp. 1687–1696, Nov. 1996.
[16] K. Venkat, L. Chen, I. Lin, P. Mistry, and P. Madhani, "Timing verification of dynamic circuits," *IEEE J. Solid-State Circuits*, vol. 31, pp. 452–455, Mar. 1996.
[17] D. Van Campenhout, T. Mudge, and K. Sakallah, "Timing verification of sequential domino circuits," in *ICCAD-96 Dig. Tech. Papers*, 1996.
[18] J. Kuskin *et al.*, "The Stanford FLASH multiprocessor," in *Proc. Int. Symp. Comp. Arch.*, Apr. 1994, pp. 302–313.

David Harris received the S.B. and M.Eng. degrees from the Massachusetts Institute of Technology, Cambridge, in 1994 and the Ph.D. degree from Stanford University, Stanford, CA, in 1999.

From 1994 to 1999, he worked in the field of high-speed integrated circuit design at Intel Corporation, Sun Microsystems, HAL Computer, and Evans and Sutherland. In 1999, he joined the faculty at Harvey Mudd College, Claremont, CA, where he is an Assistant Professor of Engineering. His research is in the field of high-speed CMOS circuit design, microprocessors, and computer graphics. He is also particularly interested in undergraduate education. He is co-author of *Logical Effort: Designing Fast CMOS Circuits* (San Francisco, CA: Morgan Kaufmann, 1999).

Mark Horowitz (S'78–M'79–SM'95) received the B.S. and M.S. degrees from the Massachusetts Institute of Technology, Cambridge, in 1978 and the Ph.D. degree from Stanford University, Stanford, CA, in 1984.

He is the Yahoo Founders Professor of Electrical Engineering and Computer Science at Stanford University. His research interests are in the area of digital systems design. He has led a umber of processor designs, including MIPS-X, one of the first processors to include on-chip instruction cache; TORCH, a statistically scheduled superscaler processor; and FLASH, a flexible DSM machine. He has also worked in a number of other chip design areas, including high-speed memory design, high-bandwidth interfaces, and fast floating point. In 1990, he took leave from Stanford University to help start Rambus, Inc., a company designing high-bandwidth memory interface technology. His current research interests include multiprocessor design, low-power circuits, memory design, and high-speed links.

Dr. Horowitz received a 1985 Presidential Young Investigator Award and an IBM Faculty Development Award, as well as the 1993 Best Paper Award at the International Solid-State Conference.

Dean Liu received the B.S. degree in electrical engineering from the University of Washington, Seattle, in 1997. He received the M.S. degree in electrical engineering from Stanford University, Stanford, CA, in 1999 where he is currently pursuing the Ph.D. degree.