Analysis and Design of Latch-Controlled Synchronous Digital Circuits

Karem A. Sakallah, *Member, IEEE,* Trevor N. Mudge, *Senior Member, IEEE,* and Oyekunle A. Olukotun

***Abstract*—We present a succinct formulation of the timing constraints for latch-controlled synchronous digital circuits. We show that the constraints are mildly nonlinear and prove the equivalence of the nonlinear optimal cycle time calculation problem to an associated and simpler linear programming (LP) problem. We present an LP-based algorithm which is guaranteed to obtain the optimal cycle time for arbitrary circuits controlled by a general class of multiphase overlapped clocks. We illustrate the formulation and an initial implementation of the algorithm on some example circuits.**

Nomenclature

k	Number of clock phases.
l	Number of latches.
ϕ_i	Clock phase i.
T_i	Duration of active interval of ϕ_i.
s_i	Start time, within the common clock cycle, of the active interval of ϕ_i.
C_{ij}	ϕ_i to ϕ_j phase-ordering flag.
K_{ij}	ϕ_i to ϕ_j input/output relationship flag.
S_{ij}	ϕ_i to ϕ_j phase-shift operator.
p_i	Clock phase used to control latch i.
A_i	Arrival time of valid data signal at input of latch i.
D_i	Departure time of valid data signal from input of latch i.
Q_i	Departure time of valid data signal from output of latch i.
Δ_{DCi}	Setup time for latch i.
Δ_{DQi}	Propagation delay of latch i.
Δ_{ij}	Propagation delay from an input latch i through a combinational logic block to an output latch j.

I. Introduction

THE analysis and design of synchronous digital circuits which are controlled by level-sensitive latches is generally acknowledged to be a difficult problem [1]-[3]. The difficulty is due mainly to the coupling between the input and output terminals of a latch while the latch is enabled. This in turn leads to a set of cyclic timing constraints which must be satisfied by a properly designed circuit. The analysis problem seeks to determine if these constraints are indeed satisfied for a given circuit and a given clocking scheme. The design problem, on the other hand, attempts to find, for a given circuit and clocking scheme, the minimum clock cycle time which would not violate these constraints. In both cases the cyclic nature of the constraints frustrates intuitive solution approaches based on simple graph traversal methods, such as CPM [4], which require the constraint set to be acyclic.

This paper has two main goals. The first is to present a succinct, yet complete, formulation of the timing constraints for latch-controlled synchronous digital circuits. The constraints in this formulation are easily constructed, almost by inspection, for any circuit topology and clocking scheme, and are clearly seen to be nonlinear, though mildly so. The second goal of the paper is to formally prove the equivalence of the nonlinear design problem (i.e., the minimum clock cycle optimization problem with nonlinear timing constraints) to an associated and simpler linear programming (LP) problem. This proof forms the basis of an LP-based algorithm for finding the optimal cycle time.

Most current methods for the analysis and design of level-sensitive synchronous digital circuits assume edge triggering to simplify the analysis and then apply some heuristics to approximate the level-sensitive constraints. As a consequence, in the analysis case, they may declare a design to be in violation of timing constraints when in fact it is not, and in the design case, they may not produce the minimum cycle time. Our modeling of the level-sensitive constraints is not an approximation and so avoids both of these problems.

This paper is organized as follows. A review of previous work in the area is given in Section II. Section III presents a formulation of the timing constraints for level-sensitive synchronous digital circuits. They are seen to be nonlinear. Section IV shows that the solution of the apparently nonlinear optimal cycle time design problem can in fact be found by solving an associated LP problem, and therefore that the entire body of LP theory can be brought to bear on design and analysis problems. An initial implementation of this LP-based solution procedure is illustrated in Section V for several example circuits. Finally, Section VI contains some concluding remarks and a discussion of future directions.

Manuscript received November 3, 1989; revised March 13, 1990. This work was supported by DARPA grant DAAL03-90-C-0028 and NSF Grant MIP-9014058. This paper was recommended by Associate Editor A. C. Parker.

K. A. Sakallah and T. N. Mudge are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

O. A. Olukotun was with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. He is now with the Computer Systems Laboratory, Stanford University, Stanford, CA.

IEEE Log Number 9104226.

II. Previous Work

The timing analysis of digital logic circuits goes back at least to the work of Kirkpatrick in the 1960's [5]. However, this and much subsequent work was concerned with timing analysis for edge-triggered logic. It has only been during this decade, with MOS VLSI emerging as the leading technology for implementing digital systems, that the timing analysis and design of level-sensitive logic have become important. In this period several authors have addressed the question of level-sensitive latches, including Agrawal [6], Jouppi [1], Ousterhout [2], Unger [7], Szymanski [8], Cherry [9], Wallace [10], and Dagenais [11], [3].

One of the earliest, Agrawal, attempted to find the maximum frequency of operation of a logic circuit through a bounded binary search algorithm. Jouppi proposed an iterative scheme based on the concept of "borrowing." In the first iteration, the critical path(s) in the circuit are determined by pretending that the latches are edge triggered. This approximation is removed in subsequent iterations. In each of these iterations an attempt is made to reduce the clock cycle time to a value determined by the second most critical path. This is accomplished by trading (borrowing) the slack time in the subcritical path. In practice, only one borrowing iteration is performed to limit the computation cost. The TV program incorporates this borrowing algorithm and has been effectively applied for the verification of several large commercial chips. Ousterhout developed Crystal, a MOS timing verification program similar in many respects to TV. However, Crystal makes no attempt at dealing with clocking issues and confines its attention to the proper modeling of signal delay through trees of MOS transistors. In fact, Ousterhout acknowledged the inherent difficulty of dealing with level-sensitive latches.

Unger developed a set of timing constraints for a limited form of two-phase clocking with level-sensitive latches. He considered both the short-path (early arrival) as well as the long-path (late arrival) problems and presented a heuristic procedure for computing the minimum cycle time subject to these constraints. This, to our knowledge, was the first explicit formulation of the timing constraints of latch-controlled circuits as a system of linear inequalities. The LEADOUT program, developed by Szymanski, is an equation-based MOS timing analysis tool which handles multiphase clocking and level-sensitive latches. The temporal behavior of latches is specified by "max" constraints similar to those encountered in CPM graphs. To eliminate the inevitable cyclic dependencies among these constraints, the circuit is first partitioned into its strongest-connected components, and constraints are generated for each cycle-free path within the components. The reference does not, however, provide enough detail about LEADOUT's clocking model to explain how the cyclic dependencies induced by clock periodicity are removed. A unique feature of LEADOUT is the compilation of the timing constraints into a fast-executing program which allows repeated analysis of a circuit with different clocking or device parameters.

More recently, several authors proposed linear programming formulations for the minimum cycle time problem. The first such formulation appears to be due to Cherry in the Pearl CMOS timing analyzer. Like LEADOUT, Pearl starts out by constructing a causality graph which captures the dependencies among data and clock signals. The data signals at the inputs and outputs of latches are then *forced* to satisfy the appropriate setup and hold times, and a set of linear inequalities which determine the "spacing" between pairs of clock edges is derived. The minimum cycle time satisfying these spacing requirements is then found by solving a linear program. The simplification of the system timing constraints to a small set of clock spacing requirements seems, however, to be based on certain implicit assumptions about when data signals at the latch inputs are available.

In ATV (Abstract Timing Verifier), Wallace introduces the notion of local and common (absolute) frames of reference for time and uses it to characterize the temporal relations among clock phases through a set of *translations*. Then, in a process similar to that used in Pearl, ATV calculates bounds on the difference between pairs of translations in the form of linear inequalities, suggesting a linear programming solution. The program, however, requires the ordering of the clock edges to be specified by the user prior to the actual optimization step. To determine the optimal clock schedule, therefore, it would be necessary to run ATV several times, with a different ordering of the clock edges each time (e.g., nonoverlapped or overlapped). Another limitation in the ATV approach is its handling of loops of level-sensitive latches. The algorithm used by the program to deal with feedback and level-sensitive latches is to unroll (replicate) the circuit graph a user-specified number of clock cycles, n_c. Beside the attendant inefficiency of such an algorithm when n_c is large, if n_c is *smaller* than the number of cycles covered by any loop of latches in the circuit, the solution generated by the program will only be an approximation to the true solution.

The most recent effort at addressing this problem is due to Dagenais. He developed a MOS timing analysis and design tool, TAMIA, which represents the timing behavior of general multi-phase-clocked latch-controlled circuits by a set of nonlinear coupled relations. The design problem, which aims at finding the optimal clock parameters for a given circuit, is then solved approximately by an iterative graph-based algorithm. Because it models a circuit at the transistor level, however, in its current implementation TAMIA carries out just one iteration of this algorithm.

III. Problem Formulation

We consider synchronous digital circuits controlled by arbitrary k-phase clocks (to be defined shortly). We assume (see Fig. 1) that the circuits can be decomposed into

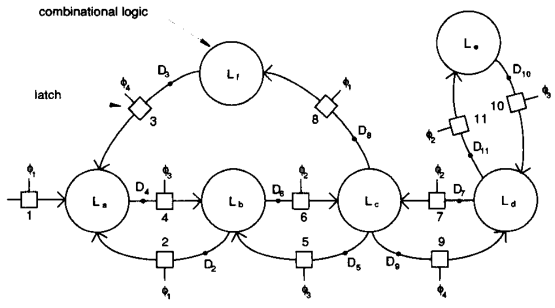


Fig. 1. Generalized logic model.

stages of feedback-free combinational logic blocks with clocked inputs and outputs.[1] The clocked elements at the inputs and outputs of the combinational stages are level-sensitive latches which provide temporary storage of data and act as *synchronizers*. These latches can be either static (for example, cross-coupled NAND gates) or dynamic (for example, MOS pass transistors). Regardless of the implementation, the functional and timing behavior of these latches is similar, and the formulation presented here applies to both types.

We place no restrictions on the combinations of clock phases used to control the input and output latches of a combinational stage other than a requirement that the set of clock phases controlling each feedback loop in the circuit be nonoverlapping; specifically, we require the logical AND of this set of phases to be identically equal to 0 at all times.

A. Clocking Methodology

An arbitrary k-phase clock is defined to be a collection of k periodic signals, $\phi_1, \phi_2, \cdots, \phi_k$, referred to as phases, with the same period. Each phase consists of two intervals: an *active* interval, during which the latches controlled by the phase are enabled; and a *passive* interval, when the latches are disabled. Without loss of generality we assume that all clock phases are active high; i.e., their active intervals occur when the phase signal assumes the logic 1 state.

We define the following clock variables (see Fig. 2):

- T_c: the clock cycle time, or period.
- s_i: the start time, relative to the beginning of the common clock cycle, of the active interval of ϕ_i.
- T_i: the duration of the active interval of ϕ_i.

For brevity, we will identify ϕ_i with its active interval, and simply refer to s_i as the start of ϕ_i and to T_i as the duration or width of ϕ_i. In addition, if ϕ_i and ϕ_j control an input latch and an output latch, respectively, of a combinational logic block L, we will simply refer to ϕ_i as an input phase, ϕ_j as an output phase, and ϕ_i/ϕ_j as an input/output-phase pair of L. We also introduce two $k \times k$ matrices C and K with elements C_{ij} and K_{ij}, defined as follows:

[1]Fig. 1 is adapted from Glasser and Dobberpuhl [13, fig. 6.7, p. 335].

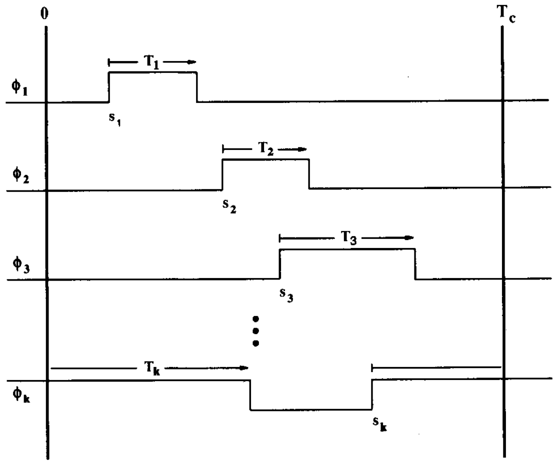


Fig. 2. Clock signal variables.

$$C_{ij} \equiv \begin{cases} 0, & i < j \\ 1, & i \geq j \end{cases} \tag{1}$$

$$K_{ij} \equiv \begin{cases} 1 & \text{if } \phi_i/\phi_j \text{ is an I/O phase pair of any logic block} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The K matrix identifies all I/O phase pairs for a particular circuit. The C matrix is used to determine if a clock cycle boundary must be crossed when going between an I/O phase pair ϕ_i/ϕ_j (see eqs. (6) and (12) below).

We can now state the relations among the various clock variables as a set of inequalities which are collectively referred to as the *clock constraints*:

C1. Periodicity Constraints:

$$T_i \leq T_c, \qquad i = 1, \cdots, k \tag{3}$$

$$s_i \leq T_c, \qquad i = 1, \cdots, k. \tag{4}$$

C2. Phase Ordering Constraints:

$$s_i \leq s_{i+1}, \qquad i = 1, \cdots, k-1. \tag{5}$$

C3. Phase Nonoverlap Constraints:

$$s_i \geq s_j + T_j - C_{ji}T_c \qquad \forall (i,j) \ni K_{ij} = 1. \tag{6}$$

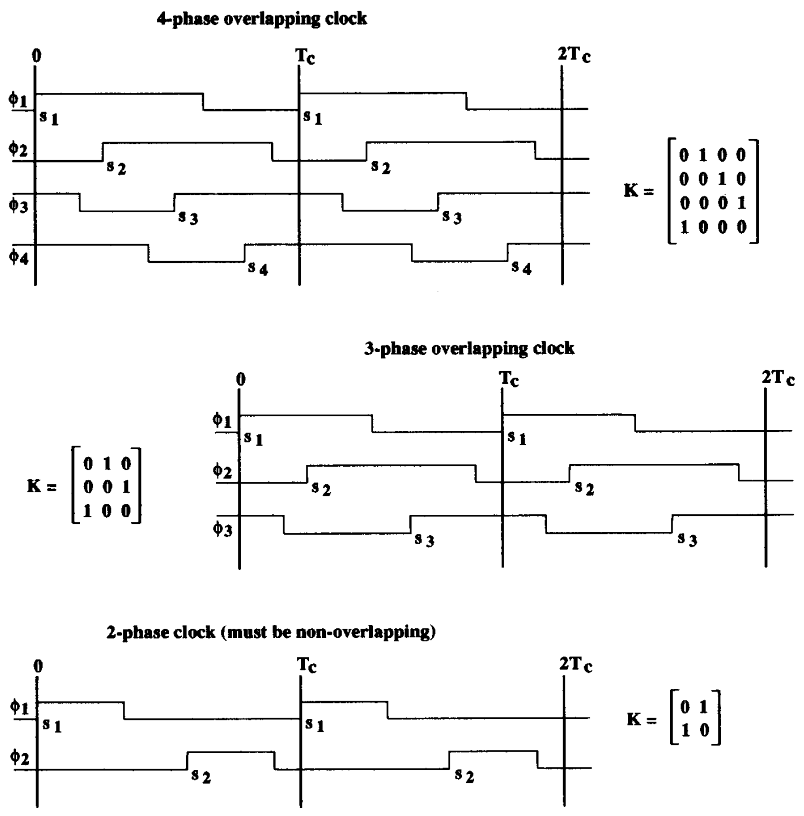
C4. Clock Nonnegativity Constraints:

$$T_c \geq 0 \tag{7}$$

$$T_i \geq 0, \qquad i = 1, \cdots, k \tag{8}$$

$$s_i \geq 0, \qquad i = 1, \cdots, k. \tag{9}$$

These inequalities, except for the nonoverlap constraints C3, are intuitively obvious. Constraints C3 ensure that the output phase ϕ_j of every I/O phase pair must end before the input phase ϕ_i starts. This in turn guarantees that the clock phases controlling any feedback loop in the circuit are never simultaneously overlapping.


4-phase overlapping clock
0
Tc
2Tc
φ1
φ2
φ3
φ4
s1
s2
s3
s4
K =
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
3-phase overlapping clock
0 1 0
0 0 1
1 0 0
2-phase clock (must be non-overlapping)
0 1
1 0
Fig. 3. Clocks with two, three, and four phases.

Constraints C1–C4 should be viewed as the minimum set of requirements that must be satisfied by a k-phase clock. Further requirements, such as minimum phase width, minimum phase separation, and clock skew, can be easily added to this minimum set but will not be treated here.

It is important to point out that the clock model introduced here is a temporal and not a logical one. The clock phases in this model are *not assumed to* have any logical relationships to one another; they are *not prevented from* being logically related either. This separation of the logical (functional) and timing aspects makes it possible to *map* any clocking discipline to our temporal framework by a suitable preprocessing step. This would include, for example, identifying derived and qualified clocks, followed by relabeling and ordering the clock phases according to (5). The generality of our clock model is demonstrated in Fig. 3 by showing how it applies to commonly used two-, three-, and four-phase clocking schemes. Note in particular that, for $k = 2$, the clock constraints ensure that the two phases are nonoverlapping, as they should be.

B. Latch Constraints

As will become evident, the simplicity of the formulation we present stems from a careful choice of time variables and naturally leads to a solution by linear programming.

We describe here the timing constraints necessary for the correct operation of D-type latches. Such latches have three terminals, representing data input, data output, and clock input (see Fig. 1). The circuit is assumed to contain l latches, numbered from 1 to l. For each of these latches we define the following variables and parameters:

- p_i: denotes the clock phase used to control latch i (e.g., latch 3 in Fig. 1 has $p_3 = 4$).
- A_i: denotes the arrival time, relative to the beginning of phase p_i, of a valid data signal at the input to latch i.
- D_i: denotes the departure time, which is the earliest time, relative to the beginning of phase p_i, when the signal available at the data input of latch i starts to propagate through the latch.
- Q_i: denotes the earliest time, relative to the beginning of phase p_i, when the signal at the data output of latch i starts to propagate through the succeeding stages of combinational logic.
- Δ_{DCi}: denotes the setup time for latch i required between the data input and the trailing edge of the clock input.
- Δ_{DQi}: denotes the propagation delay of latch i from the data input to the data output of the latch while

the clock input is high. It is assumed that $\Delta_{DQi} \geq \Delta_{DCi}$.

- Δ_{ij}: denotes the propagation delay from an input latch i through a combinational logic block to an output latch j. If latches i and j are not directly connected by a combinational block, then $\Delta_{ij} \equiv -\infty$.

Notice that both D_i and Q_i will always be nonnegative quantities, whereas A_i is unrestricted in sign.

The constraints governing latch operation fall into two categories: *setup* constraints and *propagation* constraints. The setup constraints guarantee that a latch has sufficient time to lock (store) the signal at the data input before that signal is allowed to change again. Thus,

$$A_i + \Delta_{DCi} \leq T_{p_i} \qquad i = 1, \cdots, l. \tag{10}$$

Since A_i can be negative, signifying that valid data has arrived *before* the onset of phase p_i, (10) may sometimes be satisfiable by a clock phase whose width T_{p_i} is 0! A more realistic setup constraint is obtained if A_i is replaced by D_i, yielding

$$D_i + \Delta_{DCi} \leq T_{p_i}, \qquad i = 1, \cdots, l. \tag{11}$$

In this case, since D_i is always nonnegative, the constraint places a lower bound on the width of phase p_i equal to the required setup time. We will adopt this more realistic constraint in our analysis.

Unlike the setup constraints which are local, the propagation constraints are global. They relate the departure times of signals at different latches in the circuit using the combinational propagation delay parameters. Since latch variables are referenced to the beginning of their corresponding clock phase, it is convenient to define the following phase-shift operator:

$$S_{ij} \equiv s_i - (s_j + C_{ij}T_c). \tag{12}$$

Adding S_{ij} to a time variable moves its referenced point (origin) ahead from s_i to s_j.

The propagation of signals from the inputs to the outputs of latches is simply described by

$$Q_i = D_i + \Delta_{DQi}, \qquad i = 1, \cdots, l. \tag{13}$$

Now consider a combinational path which starts at latch j and ends at latch i. The data signal at latch j starts to propagate at time Q_j and thus reaches latch i at $Q_j + \Delta_{ji}$, all times being referenced to the beginning of phase p_j. Therefore, relative to the beginning of phase p_i, the signal *arrives* at latch i at time $Q_j + \Delta_{ji} + S_{p_jp_i}$. The data signal at latch i becomes valid when *all* relevant input signals have had sufficient time to propagate through the combinational circuitry leading to latch i. Thus, the *arrival time* of a valid signal at latch i becomes

$$A_i = \max_j (Q_j + \Delta_{ji} + S_{p_jp_i}), \qquad j = 1, \cdots, l. \tag{14}$$

The propagation constraints through the combinational logic can now be expressed as follows:

$$D_i = \max (0, A_i), \qquad i = 1, \cdots, l \tag{15}$$

which express the fact that if a valid signal arrives at latch i before the start of phase p_i, then the departure time of that signal must be delayed to the beginning of phase p_i.

By eliminating the Q and A variables using (13) and (14), the latch constraints can be written exclusively in terms of signal departure times D_i along with the various clock variables, as follows:

L1. Setup Constraints:

$$D_i + \Delta_{DCi} \leq T_{p_i}, \qquad i = 1, \cdots, l. \tag{16}$$

L2. Propagation Constraints:

$$D_i = \max (0, \max_j (D_j + \Delta_{DQj} + \Delta_{ji} + S_{p_jp_i})),$$
$$i, j = 1, \cdots, l. \tag{17}$$

L3. Latch Nonnegativity Constraints:

$$D_i \geq 0, \qquad i = 1, \cdots, l. \tag{18}$$

Using the notation scheme defined in this section, it is now possible to write down the set of timing constraints for arbitrary circuits by inspection. It is assumed that the circuit has been decomposed into clocked combinational stages, and that the various delay parameters have been calculated. We illustrate this process in the Appendix for the circuit shown in Fig. 1.

IV. Optimal Clock Cycle Calculation

The minimum clock cycle time can be calculated by solving the following optimization problem, denoted by P1:

P1. Optimal Cycle Time:

Minimize	T_c
Subject to	Clock Constraints C1, C2, C3, and C4
	Latch Constraints L1, L2, and L3.

P1 is a nonlinear optimization problem. Let R_1 denote the feasible region of P1, i.e., the set of solutions to P1 which satisfy the specified constraints. The nonlinearity of P1 is due to the max functions in the latch propagation constraints L2. A linear optimization problem is obtained if these propagation constraints are *relaxed* as follows:

L2R. Relaxed Propagation Constraints:

$$D_i \geq D_j + \Delta_{DQj} + \Delta_{ji} + S_{p_jp_i}, \qquad i, j = 1, \cdots, l. \tag{19}$$

Thus, we define the following linear program:

P2. Modified Optimal Cycle Time:

Minimize	T_c
Subject to	Clock Constraints C1, C2, C3, and C4
	Latch Constraints L1, L2R, and L3

If R_2 is defined as the feasible region of P2, it should be obvious that $R_1 \subseteq R_2$.

Our objective in this section is to show that the minimum cycle time obtained by solving the linear program P2 is the same as the minimum cycle time of the original nonlinear problem P1. If we denote the optimal value for P1 and P2 by $T_{c,\min}^{(P1)}$ and $T_{c,\min}^{(P2)}$, respectively, then this result can be expressed by the following theorem:

Theorem 1:

$$T_{c,\min}^{(P1)} = T_{c,\min}^{(P2)}.$$

Proof: The critical element of this proof is the observation that the optimal value of a linear program cannot be improved with the addition of extra constraints [4, p. 170]. It folllows that, since P2 is a relaxed version of P1, $T_{c,\min}^{(P1)} \geq T_{c,\min}^{(P2)}$. Therefore the theorem is proved if we can show that P2 can be augmented with constraints such that the following two stipulations are true:

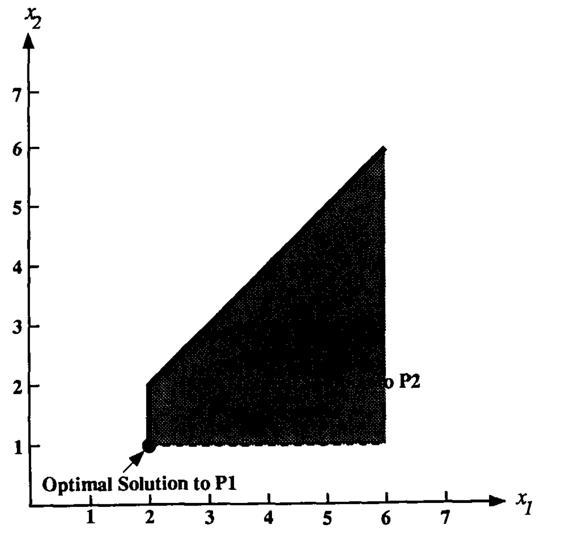
1) The optimal value of the augmented problem is not *greater* (worse) than $T_{c,\min}^{(P2)}$.
2) The constraints of the augmented problem are equivalent to those of P1.

Thus the augmented problem, which we will call P3, has the same optimal solution (i.e., the same values for all variables) as the original problem P1. Since the only difference between P1 and P2 are the latch propagation constraints, we need to examine the following two cases:

1) All D variables in the optimal solution to P2 are at their minimum values. Thus, the optimal solution to P2 satisfies all the constraints of P1, including the latch propagation constraints L2. In this case, P2 is equivalent to P1, and their optimal values are the same.
2) One or more of the D variables is not at its minimum value. Thus the optimal solution to P2 violates some of the latch propagation constraints L2. To force the solution to satisfy the L2 constraints, we augment the constraints of P2 with equality constraints as follows:
   (a) If $A_i \leq 0$ and $D_i > 0$ for some latch i, add the equality constraint $D_i = 0$.
   (b) If $A_i > 0$ and $D_i > A_i$ for some latch i, add the equality constraint $D_i = A_i$.

   The addition of these equality constraints may in some cases cause the departure time at some other latch j which previously satisfied constraints L2 to now violate them. In such cases we add further equality constraints (either (a) or (b), as appropriate) for all such affected latches, and repeat the procedure as often as necessary, until the constraints of P3 become equivalent to those of P1. It should be obvious that the addition of such equality constraints does not increase the cycle time. Thus the two stipulations stated above are true, and the theorem is proved. □

A geometric interpretation of this theorem is shown in Fig. 4. The figure clearly shows the following relationships between the original problem P1 and the modified (relaxed) problem P2:



P1:	min	$z = x_2$	**P2:**	min	$z = x_2$
	s.t.	$x_1 = \max(2, x_2)$		s.t.	$x_1 \geq 2$
					$x_1 \geq x_2$
		$x_2 \geq 1$			$x_2 \geq 1$
		$x_1, x_2 \geq 0$			$x_1, x_2 \geq 0$

Fig. 4. Geometric interpretation of Theorem 1.

- The feasible region of P1 (the two heavy line segments) is a subset of the feasible region of P2 (the shaded area, including the bordering line segments.)
- The optimal values of P1 and P2 are equal: $z_{\min} = 1$.
- P2 does not have a unique optimal solution. In fact any point on the horizontal dashed line segment is an optimal solution to P2. On the other hand, P1 has a unique optimal solution: the single point (2, 1). If X_1 and X_2 are used to denote, respectively, the optimal solution sets to P1 and P2, then the figure makes it clear that $X_1 \subseteq X_2$. This relationship, in fact, holds regardless of the uniqueness or nonuniqueness of the optimal solutions, as can be easily verified by trying other objective functions. For example, if $z = x_1$, then $X_1 = X_2 = \{(2, x_2) \mid 1 \leq x_2 \leq 2\}$; and if $z = x_1 + x_2$, then $X_1 = X_2 = \{(2, 1)\}$.
- An optimal solution to P2 may not be a feasible solution to P1. For example, the point (4, 1) is an optimal solution to P2 but is clearly infeasible for P1. The figure suggests how such a solution may be made feasible for P1: *minimize* x_1 *until it satisfies* $x_1 = \max(2, x_2)$. Applying this to the point (4, 1) yields the correct optimal solution (2, 1) to P1. The minimization step can, in fact, be carried out by "sliding" x_1 to the left until the max constraint in P1 is satisfied.

Theorem 1 forms the basis for the following algorithm to find the optimal solution of P1 by linear programming:

Algorithm MLP: Optimal Cycle Time Calculation by Modified LP

```
(Comment: m is an iteration counter; v is a convergence flag.)
1) Solve P2. Denote the optimal solution found by $D_i^0$ for $i = 1, \cdots, l$.
2) Set $m = 0$, $v =$ TRUE.
3) For $i, j = 1, \cdots, l$, evaluate $D_i^{m+1} = \max(0, \max_j (D_j^m + \Delta_{DQj} + \Delta_{ji} + S_{p_j p_i}))$.
4) If $D_i^{m+1} \neq D_i^m$ for any $i$, set $v =$ FALSE, and increment $m$.
5) If $v ==$ TRUE stop; otherwise set $v =$ TRUE and go to (3).
```

Several observations should be made about this algorithm:

- The overall performance of the algorithm depends on how efficiently we can solve the LP P2 in step 1. The most commonly used method for solving linear programs is the simplex algorithm, which, on average, takes between n and $3n$ steps to reach the optimum, where n is the number of problem constraints [4, p. 54]. For P2, it is easy to show that the number of constraints is bounded from above by $4k + (F + 1)l$, where F is the *maximum* fan-in to any latch in the circuit and is usually a small number, such as 3 or 5. Thus, the number of constraints is *linear* in the number of latches, l. Additionally, by lumping latches corresponding to vector signals with similar timing (e.g., 32-bit data buses), the number l can be reasonably small even for large circuits. The complexity of step 1, therefore, grows only linearly with l. The cost of this step can be reduced further by taking advantage of the special properties of this particular LP. For instance, by treating all latches as though they were positive-edge-triggered flip-flops, a very good initial guess can be quickly generated and used as the starting point for optimization.
- The algorithm involves a Jacobi-style iteration (steps 3 to 5) which updates the values of the departure times until all the max propagation constraints are satisfied. A more efficient Gauss-Seidel-style iteration is obviously possible. In fact, an event-driven update mechanism which only calculates the departure times which have changed from the previous iteration can be easily implemented. With such an enhancement, the cost of the iterative steps is greatly reduced for large circuits.
- When the departure times are udpated in step (3), the clock variables are held fixed at the optimal values that were found by solving the LP in step (1). Thus, the update process fixes the clock "schedule" and "slides" the departure variables toward the time origin.
- The update iteration is guaranteed to terminate because the departure times are bounded from below. In the examples we have attempted, the update process usually terminated in two to three iterations (in some cases no iterations were even necessary.)

V. Examples

In this section we illustrate our proposed formulation with three examples. The examples were solved using an initial implementation of the MLP algorithm which incorporates a simple parser, a dense-matrix LP solver which implements the standard simplex algorithm, and graphical output routines.

The first example, adapted from [3], is shown in Fig. 5. It is a simple two-stage system connected in a loop and controlled by a two-phase clock. To facilitate comparison with [3] we assume that all latches have equal setup and propagation delays of 10 ns. We also assume the same values for the combinational logic delays, except for block L_d, whose delay, Δ_{41}, will be varied to study its effect on the optimal cycle time. The resulting set of timing constraints are:

- Periodicity constraints: $T_i, s_i \le T_c$, $i = 1, 2$.
- Phase ordering constraints: $s_1 \le s_2$.
- Phase nonoverlap constraints: $s_1 \ge s_2 + T_2 - T_c$ and $s_2 \ge s_1 + T_1$.
- Latch setup constraints:
  $D_1 + 10 \le T_1 \quad D_2 + 10 \le T_2$
  $D_3 + 10 \le T_1 \quad D_4 + 10 \le T_2$.
- Latch propagation constraints:
  $D_1 = \max(0, D_4 + 10 + \Delta_{41} + s_2 - s_1 - T_c)$
  $D_2 = \max(0, D_1 + 10 + 20 + s_1 - s_2)$
  $D_3 = \max(0, D_2 + 10 + 20 + s_2 - s_1 - T_c)$
  $D_4 = \max(0, D_3 + 10 + 60 + s_1 - s_2)$.
- Nonnegativity constraints: $T_c \ge 0$; $T_{p_i}, s_{p_i} \ge 0, p_i = 1, 2$; and $D_i \ge 0$, $i = 1, \cdots, 4$.

Figs. 6 and 7 compare the results obtained by the MLP algorithm with the *null retardation in the initial phase* (NRIP) algorithm, described in [3]. The diagrams in Fig. 6 show, for each experiment, two complete cycles of the resulting clock schedule along with a "strip" which identifies the names and delay values for each of the combinational blocks and the times at which the data signals depart from each of the four latches. The shaded portions in these strips represent propagation through the latches themselves (Δ_{DQi}), whereas gaps in the strips indicate signals that arrive *earlier* than (and must thus wait for) the enabling edge of the corresponding clock phase. For example, the MLP solution for the $\Delta_{41} = 120$ ns case, shown in part (c) of the figure, has a cycle time of 140 ns with signals departing from latches 1 through 4, respectively, at 60 ns, 90 ns, 140 ns, and 210 ns. Furthermore, the input to latch 3 becomes valid at 120 ns, 20 ns earlier than the rising edge of ϕ_1; thus departure from latch 3 must wait until ϕ_1 rises at 140 ns.

These results lead to several observations:

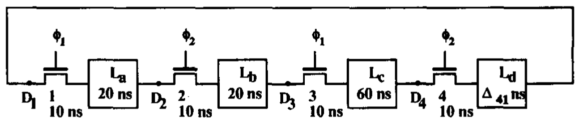


Fig. 5. Block diagram for example 1.

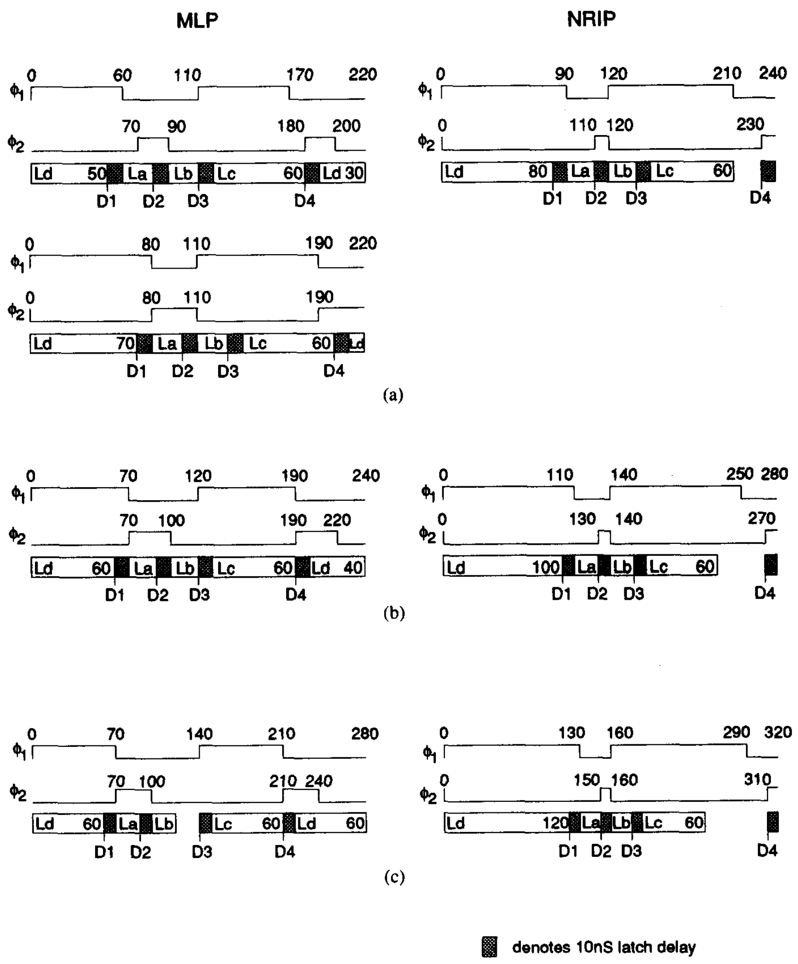


Fig. 6. Timing diagrams for example 1. (a) Δ_{41} = 80 ns. (b) Δ_{41} = 100 ns. (c) Δ_{41} = 120 ns.

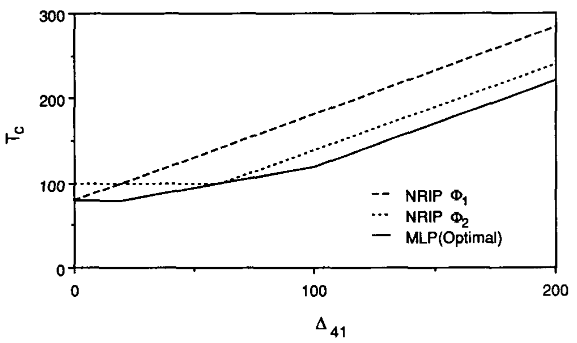


Fig. 7. T_c versus Δ_{41} for example 1.

- Unless additional constraints are placed on the minimum widths and separations of clock phases, the optimal solution will not be unique. This is illustrated with two such solutions for the Δ_{41} = 80 ns case (see the top of Fig. 6). Each of these solutions has a cycle time of 110 ns, even though their phase signals are quite different. Physically, this means that the timing constraints of a given circuit may be satisfied by a number of different clock schedules which share a common cycle time. Additional requirements, such as minimum duty cycle, may be applied to select one of these different solutions. The apparent uniqueness of the solution found by the NRIP

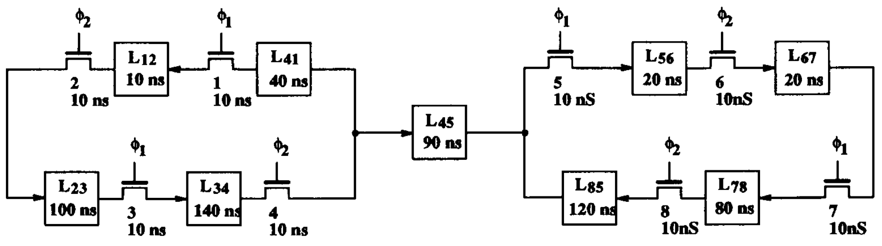


Fig. 8. Block diagram for example 2.

algorithm is due to its implicit *minimum* constraints on phase widths and separations.

- The NRIP algorithm produces an optimal solution for $\Delta_{41} = 60$ ns. For all other values of Δ_{41}, the cycle time found by NRIP is suboptimal (see Fig. 7).
- The piecewise-linear dependence of T_c on Δ_{41} in the optimal solution has three distinct segments. For $0 \leq \Delta_{41} \leq 20$ ns, T_c is independent of Δ_{41}, and is set by some other delay in the circuit. When $\Delta_{41} \geq 20$ ns, block L_d becomes critical, and any increase in Δ_{41} causes T_c to also increase. For $20 \leq \Delta_{41} \leq 100$ ns, T_c increases by 1 ns for every 2-ns increase in Δ_{41} because the added delay is shared between the two clock cycles ("borrowed" from ϕ_1). For $\Delta_{41} \geq 100$, T_c increases in direct proportion to Δ_{41}, since the additional delay can no longer be shared between the two clock cycles, and slack is inevitably introduced in the cycle with shorter delay. The rather simple dependence of the optimal cycle time on Δ_{41} in this case is due to the simplicity of the topology of this particular circuit. In fact, one can show that since the feedback loop consists of two complete clock cycles, the optimal cycle time is the maximum of the average delay around the loop and the difference between the delays for each of the cycles making up the loop.

The cycle time calculations using the MLP and NRIP algorithms for a more complicated example are shown in Figs. 8 and 9.

The following additional observations can be made:

- Unlike the previous example, the cycle time found by the NRIP algorithm is significantly higher (35%) than the optimal cycle time. While this result cannot be generalized for other circuits, it does point out that the approximate solution found by NRIP may deviate appreciably from the exact solution, and additional iterations might be necessary. Because it can be found fairly quickly, however, the NRIP solution may be used in the LP step of the MLP algorithm as a starting point.
- Because of the coupling of the timing constraints through the feedback loops in the circuit as well as through the periodic clock signals, the notion of a *critical path* is clearly inadequate as a basis for discussing the optimality of the solution, and the dependence of that solution on the circuit's delays. Instead of a single critical path, the circuit has several critical combinational delay *segments* which may be disjoint. The criticality of these segments, and the subcriticality of others, are directly related to associated slack variables in the inequality constraints. The techniques of parametric analysis in linear programming can be usefully applied here to study the effects of varying the circuit delays on the optimal cycle time.

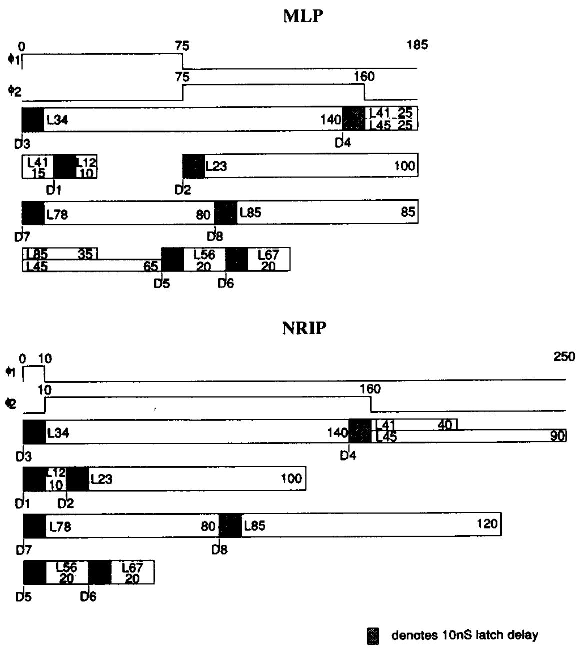


Fig. 9. Timing diagrams for example 2.

The final example illustrates the application of the MLP algorithm to study and optimize the timing of a 250 MHz gallium arsenide microcomputer currently under development at the University of Michigan [12]. Fig. 10 shows a simplified block diagram of the microcomputer's CPU and its primary cache subsystem. The CPU implements an existing instruction-set architecture, the MIPS R6000, and has as its main components a register file of 32 32-bit registers, an ALU, a shifter, and an integer multiply and divide unit. The data path is 32 bits wide and is timed

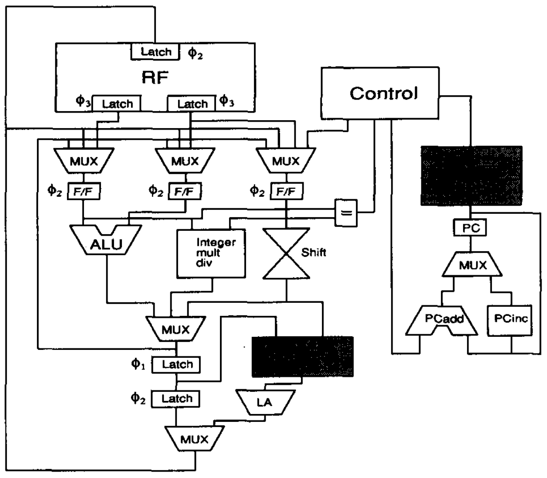


Fig. 10. GaAs MIPS system diagram.

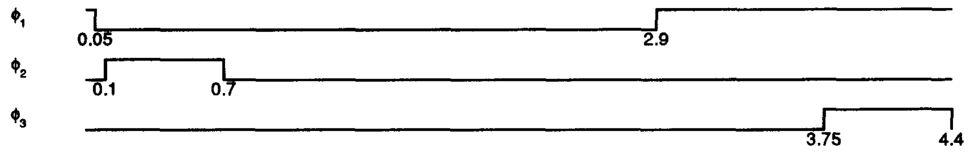


Fig. 11. MLP-generated clock schedule for the GaAs MIPS data path.

with a three-phase clock. The diagram shows the major logic blocks in the CPU data path, which is implemented as a single chip. The shaded blocks show the instruction cache and the data cache, which are implemented as a set of 15 high-speed GaAs 1 K $\times$ 32 SRAM chips. The synchronizing elements are a combination of latches and flip-flops (F/F's, see Fig. 10). To reduce the effects of chip crossings the CPU and the primary caches are integrated into a single multichip module (MCM).

We applied the MLP algorithm to a timing model of the data path to obtain its optimal clock schedule, assuming that the cache subsystem could be designed to match the speed of the CPU. The data path contains roughly 30 000 transistors, the majority of which are in the register file (see Table I for a breakdown of the transistor count amoung the major data path blocks). The timing model was abstracted from the transistor level, and consists of 18 synchronizing elements, 15 of which are level-sensitive latches. Each of the synchronizers in the model represents a 32-bit-wide data bus. The timing parameters (propagation delays and setup times) were extracted from circuit simulations using SPICE. The resulting clock schedule is shown in Fig. 11. It is interesting to note the following:

- The number of constraints for this example was 91. Even though the current implementation of the MLP algorithm uses a dense-matrix solver, its execution time (on a DECStation 3100) was hardly noticeable (on the order of a few seconds).

TABLE I
TRANSISTOR COUNT FOR MAJOR BLOCKS OF THE GaAs MIPS DATAPATH

Block Name	No. of Transistors
Register File (RF)	16,085
Arithmetic/Logic Unit (ALU)	3419
Shifter	1848
Integer Multiply/Divide (IMD)	6874
Load Aligner	1922
Total	30,148

- The optimal cycle time found by MLP (4.4 ns) is 10% higher than the target cycle time of 4 ns. We are continuing to refine the delay parameters of the model from additional circuit simulations as well as actual measurements on prototype chips, and to apply the MLP algorithm throughout the design process in order to monitor any changes in the optimal cycle time.
- Phase ϕ_3 in the optimal clock schedule is completely overlapped by ϕ_1. While this relationship might seem odd at first, it is easy to explain once the function of ϕ_3 in the circuit is recognized. This third phase is used as a precharge clock for the register file storage cells to speed up the readout of data into the ALU. The total overlap of ϕ_3 by ϕ_1 is not a problem since there are no direct paths in the circuit between these two phases (i.e., $K_{13} = K_{31} = 0$). This result also

points out some of the generality of the timing model proposed in this paper, namely that it is able to overlap clock phases if necessary to produce a shorter cycle time.

VI. Conclusions

We have shown in this paper that the optimal cycle time for circuits controlled by level-sensitive latches can be determined by solving a linear program. In contrast to earlier models, the timing constraints presented here are quite general, being based on very few assumptions, allowing them to accommodate a much wider class of clocking schemes. In addition, these constraints are very easy to generate for arbitrary circuit structures. The LP formulation provides a convenient theoretical foundation for analyzing such constraints and for developing algorithms that are potentially more efficient than the simplex algorithm. We are currently investigating just such algorithms, noting that the entries of the constraint matrix for this problem are exclusively topological (i.e., 0, ± 1). We also intend to use parametric programming techniques to quantify the notion of critical path segments and to study the effects on the optimal cycle time of varying the circuit delays.

Appendix

To illustrate the notation developed in Section III, we present here the complete set of timing constraints for the circuit shown in Fig. 1. The circuit has 11 latches and is controlled by a four-phase clock with the following K matrix:

$$K = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

Thus there are nine I/O phase pairs; the corresponding phase-shift operators (used in the latch propagation constraints) are

$$S_{13} = s_1 - s_3$$
$$S_{14} = s_1 - s_4$$
$$S_{21} = s_2 - s_1 - T_c$$
$$S_{23} = s_2 - s_3$$
$$S_{24} = s_2 - s_4$$
$$S_{31} = s_3 - s_1 - T_c$$
$$S_{32} = s_3 - s_2 - T_c$$
$$S_{42} = s_4 - s_2 - T_c$$
$$S_{43} = s_4 - s_3 - T_c.$$

The timing constraints can now be stated as follows:

- Periodicity constraints:

$$T_i \le T_c,\ s_i \le T_c, \qquad i = 1, 2, 3, 4.$$

- Phase-ordering constraints:

$$s_1 \le s_2 \le s_3 \le s_4.$$

- Phase nonoverlap constraints:

$$s_1 \ge s_3 + T_3 - T_c \qquad s_2 \ge s_1 + T_1 \qquad s_3 \ge s_1 + T_1$$
$$s_4 \ge s_2 + T_2 \qquad s_1 \ge s_4 + T_4 - T_c$$
$$s_2 \ge s_3 + T_3 - T_c$$
$$s_3 \ge s_2 + T_2 \qquad s_4 \ge s_3 + T_3 \qquad s_2 \ge s_4 + T_4 - T_c.$$

- Latch setup constraints:

$$D_i + \Delta_{DCi} \le T_1, \qquad i = 1, 2, 8$$
$$D_i + \Delta_{DCi} \le T_2, \qquad i = 6, 7, 11$$
$$D_i + \Delta_{DCi} \le T_3, \qquad i = 4, 5, 10$$
$$D_i + \Delta_{DCi} \le T_4, \qquad i = 3, 9.$$

- Latch propagation constraints:

$$D_2 = \max(0, D_4 + \Delta_{DQ4} + \Delta_{42} + S_{31}, D_5 + \Delta_{DQ5} + \Delta_{52} + S_{31})$$
$$D_3 = \max(0, D_8 + \Delta_{DQ8} + \Delta_{83} + S_{14})$$
$$D_4 = \max(0, D_1 + \Delta_{DQ1} + \Delta_{14} + S_{13}, D_2 + \Delta_{DQ2} + \Delta_{24} + S_{13}, D_3 + \Delta_{DQ3} + \Delta_{34} + S_{43})$$
$$D_5 = \max(0, D_6 + \Delta_{DQ6} + \Delta_{65} + S_{23}, D_7 + \Delta_{DQ7} + \Delta_{75} + S_{23})$$
$$D_6 = \max(0, D_4 + \Delta_{DQ4} + \Delta_{46} + S_{32}, D_5 + \Delta_{DQ5} + \Delta_{56} + S_{32})$$
$$D_7 = \max(0, D_9 + \Delta_{DQ9} + \Delta_{97} + S_{42}, D_{10} + \Delta_{DQ10} + \Delta_{10,7} + S_{32})$$
$$D_8 = \max(0, D_6 + \Delta_{DQ6} + \Delta_{68} + S_{21}, D_7 + \Delta_{DQ7} + \Delta_{78} + S_{21})$$
$$D_9 = \max(0, D_6 + \Delta_{DQ6} + \Delta_{69} + S_{24}, D_7 + \Delta_{DQ7} + \Delta_{79} + S_{24})$$
$$D_{10} = \max(0, D_{11} + \Delta_{DQ11} + \Delta_{11,10} + S_{23})$$
$$D_{11} = \max(0, D_9 + \Delta_{DQ9} + \Delta_{9,11} + S_{42}, D_{10} + \Delta_{DQ10} + \Delta_{10,11} + S_{32}).$$

- Nonnegativity constraints:

$$T_c \ge 0$$
$$T_{p_i} \ge 0,\ s_{p_i} \ge 0, \qquad p_i = 1, 2, 3, 4$$
$$D_i \ge 0, \qquad i = 1, \cdots, 11.$$

References

[1] N. P. Jouppi, "Timing verification and performance improvement of MOS VLSI designs," Ph.D. thesis, Stanford University, Stanford, CA 94305-2192, Oct. 1984.

[2] J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 3, pp. 336-349, 1985.

[3] M. R. Dagenais and N. C. Rumin, "On the calculation of optimal clocking parameters in synchronous circuits with level-sensitive latches," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 268-278, Mar. 1989.

[4] D. T. Phillips, A. Ravindran, and J. J. Solberg, *Operations Research: Principles and Practice.* New York: Wiley, 1976.

[5] T. I. Kirkpatrick and N. R. Clark, "PERT as an aid to logic design," *IBM J. Res. Develop.*, vol. 10, no. 2, pp. 135-141, Mar. 1966.

[6] V. D. Agrawal, "Synchronous path analysis in MOS circuit simulator," in *Proc. 19th Design Automat. Conf.*, 1982, pp. 629-635.

[7] S. H. Unger and C.-J. Tan, "Clocking schemes for high-speed digital systems," *IEEE Trans. Comput.*, vol. C-35, pp. 880-895, Oct. 1986.

[8] T. G. Szymanski, "LEADOUT: A static timing analyzer for MOS circuits," in *ICCAD-86 Dig. Tech. Papers*, 1986, pp. 130-133.

[9] J. J. Cherry, "Pearl: A CMOS timing analyzer," in *Proc. 25th Design Automat. Conf.*, 1988, pp. 148-153.

[10] D. E. Wallace and C. H. Sequin, "ATV: An abstract timing verifier," in *Proc. 25th Design Automat. Conf.*, 1988, pp. 154-159.

[11] M. R. Dagenais, "Timing analysis for MOSFETs; An integrated approach," Ph.D. thesis, McGill University, Montreal, Quebec, Canada H3A 2A7, June 1987.

[12] R. B. Brown, J. A. Dykstra, T. N. Mudge, and R. Milano, "A GaAs microsupercomputer: Rationale and design," Tech. Rep. CSE-TR-42-90, University of Michigan, Dept of EECS, Ann Arbor, MI 48109-2122, 1980.

[13] L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits.* Reading, MA: Addison Wesley, 1985.

Karem A. Sakallah (S'76-M'81) received the B.E. degree (with distinction) in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1975 and the M.S.E.E. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1977 and 1981, respectively.

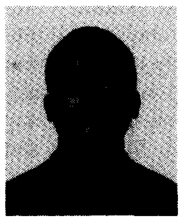
In 1981 he joined the Department of Electrical Engineering at CMU as a Visiting Assistant Professor. From 1982 to 1988 he was with the Semiconductor Engineering Computer-Aided Design Group at the Digital Equipment Corporation, Hudson, MA, where he headed the Analysis and Simulation Advanced Development team. Since September 1988 he has been at the University of Michigan, Ann Arbor, as Associate Professor of Electrical Engineering and Computer Science. His research interests are primarily in the area of computer-aided design of integrated circuits and systems, with particular emphasis on numerical analysis, multilevel simulation, timing verification and optimal clocking, modeling, knowledge abstraction, and design environments.

Dr. Sakallah is a member of the ACM.

Trevor N. Mudge (S'74-M'77-SM'84) received the B.Sc. degree in cybernetics from the University of Reading, England, in 1969, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively. While at the University of Illinois he participated in the design of several special-purpose computers, and did research in computer architecture.

Since 1977, he has been on the faculty of the University of Michigan, Ann Arbor, where he has taught classes on logic design, CAD, computer architecture, and programming languages. He is presently the Director of the Advanced Computer Architecture Lab and Associate Professor of Electrical Engineering and Computer Science.

Dr. Mudge is the author of more than 100 papers on computer architecture, programming languages, VLSI design, and computer vision, and he holds a patent in computer-aided design of VLSI circuits. His major research, at present, is the design and construction of a high-performance GaAs microcomputer. In addition to his position as a faculty member, he is a consultant for several computer companies in the areas of architecture and languges. Dr. Mudge is a member of the ACM and of the British Compuer Society.



Oyekunle A. Olukotun received the B.S. degree in electrical engineering in 1985 and the M.S. degree in computer engineering in 1987 from the University of Michigan. Currently, he is pursuing the Ph.D., also at the University of Michigan. His research interests include parallel algorithms for computer-aided design of integrated circuits and tools for the analysis, design, and verification of high-performance digital systems.