Optimal Clocking of Synchronous Systems

Karem A. Sakallah, Trevor N. Mudge and Oyekunle A. Olukotun
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

July 30, 1990 @ 10:23

Abstract

The need for accurate characterization of the timing behavior of synchronous digital systems has become increasingly evident in the past few years. The continuing drive to design faster digital systems has made the use of CAD tools for timing verification and optimal clocking essential in most system design methodologies. Timing models of synchronous digital circuits are based on the premise that the circuits are logically (functionally) correct, and focus only on capturing their propagation, latching, and synchronization properties. While, in principle, such timing models are much simpler than full-fledged logical models, the increasing use of multi-phase clocking schemes and level-sensitive latching structures has significantly increased their complexity.

In this paper we introduce a new timing model of synchronous digital circuits which is: 1) general enough to handle arbitrary multi-phase clocking; 2) complete, in the sense that it captures signal propagation along short as well as long paths in the logic; 3) extensible to make it relatively easy to incorporate "complex" latching structures; and 4) notationally simple to make it amenable to analytic treatment in some important special cases.

1 Introduction

This paper introduces a *timing model* of synchrounous digital circuits and describes its application to find the *optimal (maximum-rate) clocking schedules*. The model assumes that the circuits are logically (functionally) correct, and focuses only on capturing their propagation, latching, and synchronization properties. It can be viewed as a synthesis of ideas from earlier work in the field of timing analysis of synchronous systems. On the other hand, it also includes several key concepts which make it—as will become evident—significantly more versatile than previous models. In particular, on the issue of level-sensitive latches which has received a great deal of attention recently [1]–[10], the model offers a general and accurate, yet simple, treatment.

Briefly, the model can be characterized by the following:

- A clear distinction between *data* and *clock* signals.
- A general treatment of *multi-phase* clocks which underscores the centrality of the common clock cycle T_c and emphasizes the *temporal* rather than the *logical* relations among clock *phases*; in particular, the notions of phase-relative *time zones* and of a *phase-shift operator* eliminate much of the notational clutter which has plagued previous efforts in dealing with "complex" clocking schemes.
- An extensible framework which allows the incorporation of arbitrary *synchronizing* structures, i.e. circuit structures where clock and data signals converge and interact, as long as a timing *macromodel* of these interactions can be provided; in this paper we present timing macromodels for D-type latches and flip-flops.
- Completeness, in the sense that the model accounts for signal propagation along the *shortest* as well as the *longest* paths in a circuit.

and, finally,

- Simplicity, through careful attention to notation and the use of variable and parameter symbols which have mnemonic value.

The model presented here extends our earlier model in [11] to handle propagation along short paths. In fact, aside from some minor changes in notation, the "new" model is a superset of the earlier one (see Sec. 4). We have implemented the model in two prototype CAD tools: *check*T_c which examines a circuit for adherence to a specified clock schedule, and reports on setup and hold time violations; and *min*T_c which determines the optimal clock schedule (i.e. the schedule with the minimum cycle time) that satisfies all the timing constraints for a given circuit. The remainder of the paper is organized as follows. In Sec. 2 we present a brief review of previous work. The formulation of the proposed timing model is developed in Sec. 3. In Sec. 4 we introduce worst-case assumptions about short-path propagation and derive a simpler, albeit more restrictive, alternative to the general model. An example illustrating the application of both models is given in Sec. 6, and we close with some conclusions and suggestions for future work in Sec. 7.

2 Previous Work

The timing analysis of digital logic circuits goes back at least to the work of Kirkpatrick in the 1960's [12]. Since then, a great deal of effort has been devoted to the subject. However, much of

this work, including Kirkpatrick's original paper, was concerned with timing analysis for edge-triggered logic employing simple clocking methodologies. The timing models for this class of circuits are fairly simple because edge-trigerring decouples the timing of flip-flop outputs from the timing of their inputs. In contrast, the timing models for circuits employing level-sensitive latches are coupled and considerably more complex; studies of latch-controlled circuits during that period were, therefore, mostly limited to regular circuit topologies, such as pipelines [13]. It has only been during the last ten years, with MOS VLSI emerging as the leading technology for implementing digital systems, that the timing analysis and design of level-sensitive logic with sophisticated clocking schemes has become important. In this period several authors have addressed the question of level-sensitive latches and multi-phase clocking. We review here the work of McWilliams [14], Agrawal [15], Jouppi [1], Ousterhout [2], Glesner [3], Unger [4], Szymanski [5], Cherry [6], Wallace [7], Ishiura [8], Weiner [9], and Dagenais [10].

McWilliams' SCALD system is a simulation-based event driven timing verifier which handles both latches and flip-flops. Using a seven-valued algebra, it simulates the circuit over one complete clock cycle, and reports violations, at storage elements, of setup and hold times, as well as minimum required clock pulse widths. Clocks must be specified, but they can be quite arbitrary. When the circuit contains latches, convergence to the correct signal waveforms requires several simulation "passes." The work by Agrawal, one of the earliest to address the design problem, attempts 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 from the sub-critical path. In practice, only one borrowing iteration is performed to limit the computation cost. The TV timing verifier 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. Glesner's SCAT program uses an iterative algorithm to calculate the minimum cycle time in the context of a 2-point statistical delay model.

Unger developed a set of timing constraints for a limited form of 2-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 general latch-controlled circuits as a system of linear inequalities, although linear inequalities were used earlier by Davidson and Fawcett [13] to model the timing of pipelines.

The LEADOUT program, developed by Szymanski, is an equation-based MOS timing analysis tool which handles multi-phase 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 make it clear, however, 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.

Several authors formulated the minimum cycle time problem as a linear program. The first such formulation appears to be due to Cherry in the Pearl CMOS timing analyzer. Like LEAD-OUT, 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 differences between pairs of translations in the form of linear inequalities, suggesting a linear programming solution. Ishiura's *time-symbolic* simulation similarly introduces time variables to characterize events, and relates them with linear inequalities which model timing constraints in asynchronous logic circuits. The constraints are then solved as a linear program to determine their feasibility, i.e. to determine if the circuit is free from hazards.

In contrast with the linear programming approach, several authors recently introduced new graph-based algorithms to solve for the minimum cycle time. Dagenais, for example, 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 a relaxation-type algorithm which decouples the timing relations during each iteration. The TAMIA program is based on a single-iteration implementation of this algorithm, referred to as the *null retardation in the initial phase*. Weiner's Hummingbird program is a timing analyzer used in the context of a logic synthesis environment. It offers a very general model of clocking, and handles level-sensitive latches. The program incorporates several iterative algorithms for determining the slow paths in a design using the notions of "slack transfer" and "time snatching". It does not, however, model the short-path problem.

3 General System Timing Constraints

Our timing model is based on making a distinction between two types of signals in a synchronous circuit: *clock* signals and *data* signals. Clock signals are used to *regulate* the flow of data signals in the circuit to make sure that no data signal changes any sooner or any later than it is supposed to. This *synchronizing* action of clock signals is predicated on precisely knowing the value of a clock signal at all instants during a clock cycle[1]. The values of data signals, on the other hand, need not be known precisely; it is sufficient for timing purposes to merely know when a data signal is *stable* and when it is *changing* during a clock cycle. Much of the power of timing models, over logical models, is due to this *data-independence* which effectively achieves complete coverage of *all* signal propagation paths in a circuit without having to specify test vectors at the

[1]Some uncertainty in the value of a clock signal is, however, unaviodable because of finite rise and fall times.

circuit inputs[2].

The distinction between data and clocks leads naturally to three categories of timing relations in a synchronous circuit:

1. Relations among different clock signals, or phases, of a periodic clock.
2. Relations among data and clock signals for various types of synchronizing elements.
3. Relations among different data signals at the inputs and outputs of combinational logic blocks.

The remainder of this section is devoted to the development of each of these three categories of timing relations. Collectively, these relations will be referred to as the *General System Timing Constraints (GSTC).*

3.1 Clocking Model

We define an arbitrary k-phase clock to be a collection of k periodic signals ϕ_1 , ϕ_2 , $\cdots$, ϕ_k — referred to as *phases* — with a common cycle time T_c. The phase signals are applied to the control inputs of synchronizing elements, such as latches and flip-flops. Each phase divides the clock cycle into two intervals: an *active* interval of duration T_i, and a *passive* interval of duration $(T_c - T_i)$. During the active interval of a given phase, the synchronizers it controls are *enabled*; during its passive interval, they are *disabled.* The transitions *into* and *out of* the active interval will be called, respectively, the *enabling* and *latching* edges of the phase. We assume, without loss of generality, that all phases are active high; thus, the enabling and latching edges correspond to the rising and falling transitions of the phase signal. It is important to point out that the phase signals are *not* required to be non-overlapping. In fact, as we will see in Sec. 6, shorter cycle times can be obtained when the clock phases are allowed to overlap.

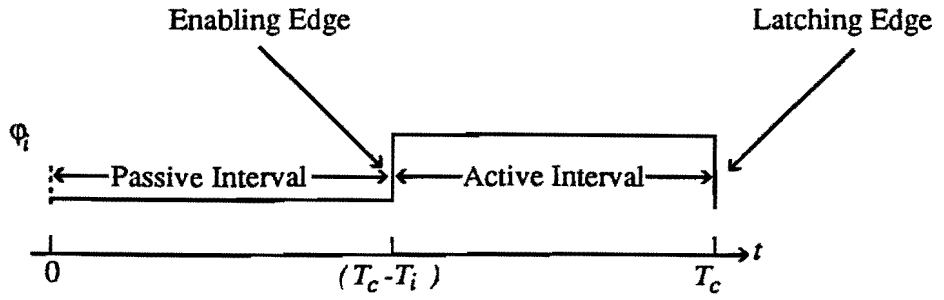


Figure 1: Clock phase ϕ_i and its local time zone

Associated with each phase is a *local time zone*, as shown in Fig. 1, such that its passive interval starts at $t = 0$, its enabling edge occurs at $t = T_c - T_i$, and its latching edge occurs at $t = T_c$. Mathematically, the domain of the local time zone is defined to be the interval $(0, T_c]$ since the start of the current clock cycle coincides with the end of the previous cycle. The temporal relationships among the k phases (i.e. among the different time zones) can now be established by

[2]Of course, some logically-impossible paths may also be covered necessitating the inclusion of some degree of data-dependence [1, 2].

an arbitrary choice of a *global time reference*. We introduce e_i to denote the time, relative to this global time reference, at which phase ϕ_i *ends* (i.e. when its latching edge occurs). We also assume that the phases are *ordered*[3] in this global time reference so that $e_1 \leq e_2 \leq \cdots \leq e_{k-1} \leq e_k$. The global time reference is now *arbitrarily* chosen to coincide with the local time zone of ϕ_k; thus $e_k \equiv T_c$.

Finally, we define a *phase shift* operator:

$$E_{ij} \equiv \begin{cases} (e_j - e_i)\,, & i < j \\ (T_c + e_j - e_i)\,, & i \geq j \end{cases} \tag{1}$$

which takes on positive values in the range $[0, T_c]$. When subtracted from a timing variable in the *current* local time zone of ϕ_i, E_{ij} changes the frame of reference to the *next* local time zone of ϕ_j, taking into account a possible cycle boundary crossing.

3.2 Synchronizer Model

We develop here the timing relations for D-type synchronizers, either latches or flip-flops[4] with three terminals each: data input, data output, and clock input. The latches can be either static (for example cross-coupled NAND gates) or dynamic (for example MOS pass transistors). The circuit is assumed to contain l synchronizers numbered from 1 to l. The ith synchronizer is characterized by the following five parameters:

- p_i: clock phase used to control synchronizer i.
- S_i: setup time of synchronizer i relative to latching edge of p_i.
- H_i: hold time of synchronizer i relative to latching edge of p_i.
- δ_i , Δ_i: minimum and maximum propagation delay of synchronizer i; for latches, propagation is from the data input to the data output; for flip-flops, propagation is from the clock input to the data output.

For timing purposes, it is sufficient to characterize a data signal over one clock cycle by two, possibly simultaneous, events which demark the interval when the signal is switching between its old and new values. For the signal *arriving* at the data input of synchronizer i these two events are defined to occur at $t = a_i$ and $t = A_i$ in the local time zone of phase p_i. The corresponding events of the data signal *departing* from the synchronizer are defined to occur at $t = d_i$ and $t = D_i$. It will be convenient to refer to a_i and A_i as the *early* and *late* arrival times, and to d_i and D_i as the *early* and *late* departure times. The synchronizing action can now be completely specified in terms of these four event times as follows (see Fig. 2):

- Latch Synchronization:

$$\begin{aligned} d_i &= \max(a_i, T_c - T_{p_i}) \\ D_i &= \max(A_i, T_c - T_{p_i}) \end{aligned}$$

- Flip-flop Synchronization:

$$d_i = D_i = T_c$$

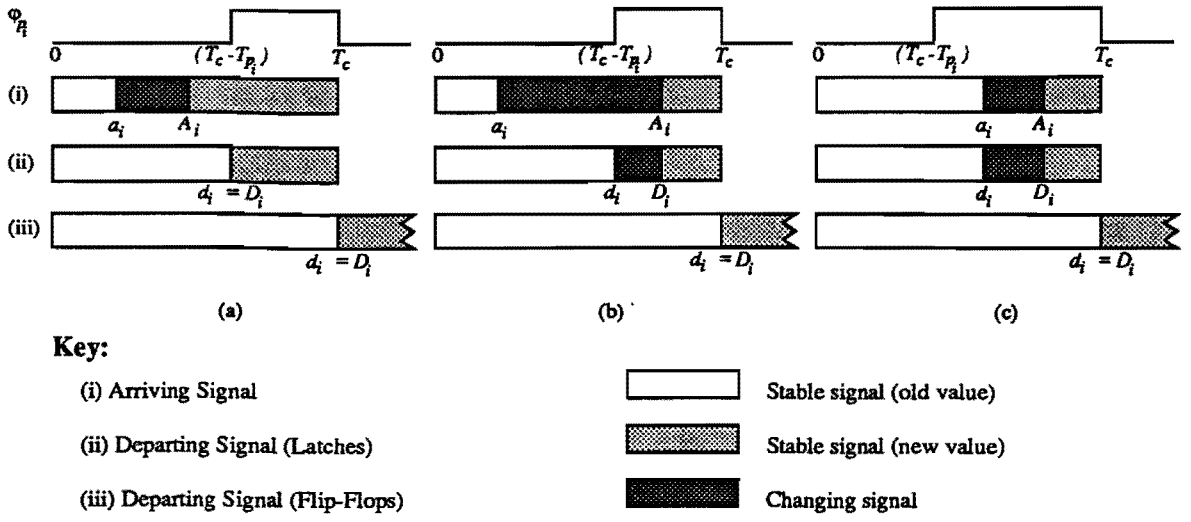


Figure 2: Synchronizing Action of Latches and Flip-Flops

The above equations clearly illustrate the difference between the operation of latches and flip-flops: for flip-flops, the timing of the departing signal is *independent* of the timing of the arriving signal; for latches, the timing of the departing signal is *dependent* on the timing of the arriving signal. It is this dependence which causes much of the complexity in dealing with latch-controlled circuits.

For correct operation, the arriving data signal must satisfy the following hold and setup time requirements (Fig. 3):

$$\begin{aligned} a_i &\geq H_i \\ A_i &\leq T_c - S_i \end{aligned}$$

3.3 Combinational Logic Model

The combinational logic in the circuit is assumed to have been decomposed into *stages* with clocked inputs and outputs[5] (see Fig. 4), and is characterized for timing purposes by two $l \times l$ matrices whose elements are defined as follows:

- δ_{ij}: minimum propagation delay from the data output of synchronizer i through a combinational logic stage to the data input of synchronizer j; if synchronizers i and j are not directly connected by combinational logic, then $\delta_{ij} \equiv \infty$.
- Δ_{ij}: maximum propagation delay from the data output of synchronizer i through a combinational logic stage to the data input of synchronizer j; if synchronizers i and j are not directly connected by combinational logic, then $\Delta_{ij} \equiv -\infty$.

[3]This ordering does not imply any restrictions on clocking; it is merely a labeling device for notational convenience.

[4]The model described here applies to negative edge-triggered flip-flops. Models for positive edge-triggered and master-slave flips-flops can be defined similarly.

[5]Figure 4 is adapted from [16, Fig. 6.7 p. 335].

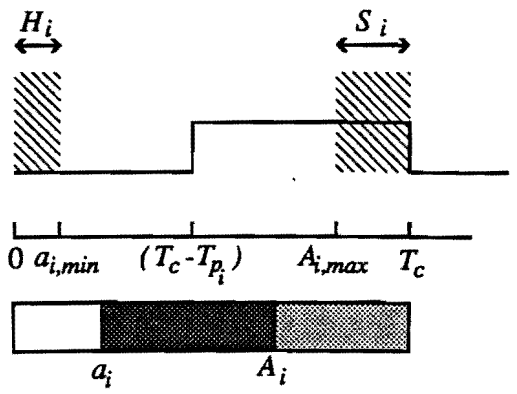


Figure 3: Hold and setup time constraints

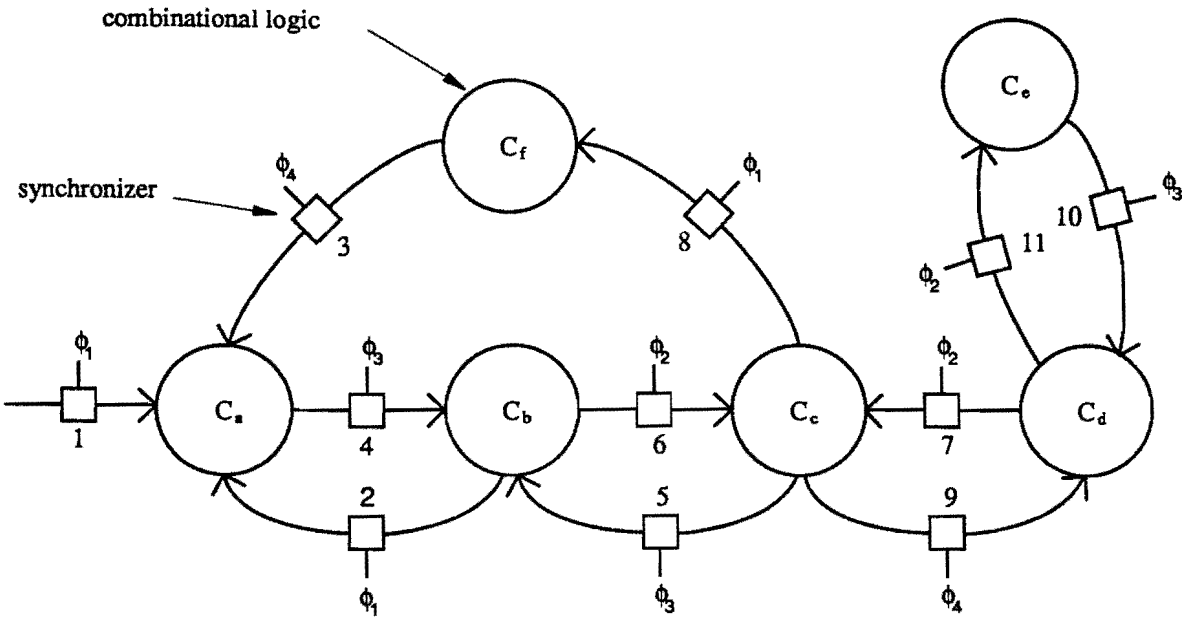


Figure 4: Generalized Synchronous Circuit Model

Both matrices δ and Δ can be simultaneously obtained by a breadth-first or a depth-first critical path algorithm applied to each combinational stage in the circuit[17, Ch. 3].

The temporal model of the combinational logic can now be stated by the following two equations:

$$\begin{aligned} a_i &= \min_j(d_j + \delta_j + \delta_{ji} - E_{p_jp_i}) \qquad i,j = 1,\cdots,l \\ A_i &= \max_j(D_j + \Delta_j + \Delta_{ji} - E_{p_jp_i}) \quad i,j = 1,\cdots,l \end{aligned}$$

These two equations express the two arrival time events at synchronizer i in terms of the corresponding departure time events at its *fan-in* synchronizers j. For example, in Fig. 4, $A_7 = \max(D_9 + \Delta_9 + \Delta_{97} - E_{42}, D_{10} + \Delta_{10} + \Delta_{10,7} - E_{32})$. The subtraction of the phase shift operators E_{42} and E_{32} achieves the necessary and unifying change of reference from the local time zones of phases ϕ_4 and ϕ_3 to that of phase ϕ_2.

3.4 Summary

The complete set of general system timing constraints, GSTC, is summarized in Table 1. For ease of reference, the constraints have been organized into five categories, and have been given mnemonic labels.

GC1. Phase Ordering Constraints:	$e_{i-1} \le e_i$	$i = 2,\cdots,k$
	$e_k \equiv T_c$	
GC2. Latching Constraints:		
GC2H. Hold Time Constraints:	$a_i \ge H_i$	$i = 1,\cdots,l$
GC2S. Setup Time Constraints:	$A_i \le T_c - S_i$	$i = 1,\cdots,l$
GC3. Synchronization Constraints:		
GC3L. Latch Synchronization:		
GC3Ld. Early Departure:	$d_i = \max(a_i, T_c - T_{p_i})$	$i = 1,\cdots,l$
GC3LD. Late Departure:	$D_i = \max(A_i, T_c - T_{p_i})$	$i = 1,\cdots,l$
GC3F. Flip-Flop Synchronization:		
GC3Fd. Early Departure:	$d_i = T_c$	$i = 1,\cdots,l$
GC3FD. Late Departure:	$D_i = T_c$	$i = 1,\cdots,l$
GC4. Combinational Propagation Constraints:		
GC4a. Early Arrival:	$a_i = \min_j(d_j + \delta_j + \delta_{ji} - E_{p_jp_i})$	$i,j = 1,\cdots,l$
GC4A. Late Arrival:	$A_i = \max_j(D_j + \Delta_j + \Delta_{ji} - E_{p_jp_i})$	$i,j = 1,\cdots,l$
GC5. Bound Constraints:	$0 \le e_i\ ,\ T_i \le T_c$	$i = 1,\cdots,k$
	$0 \le a_i\ ,\ A_i\ ,\ d_i\ ,\ D_i \le T_c$	$i = 1,\cdots,l$

Table 1: General System Timing Constraints (GSTC)

4 Restricted System Timing Constraints

Constraints GC1-GC5 are complete in the sense that they model propagation along both the shortest and longest paths, and insure that hold and setup time requirements are satisfied at all synchronizers. A frequently made simplification is to assume that the minimum propagation delays in the circuit are zero. This assumption makes it possible to replace the short-path constraints by a smaller set of phase non-overlap constraints. This can be readily seen for latch-type synchronizers

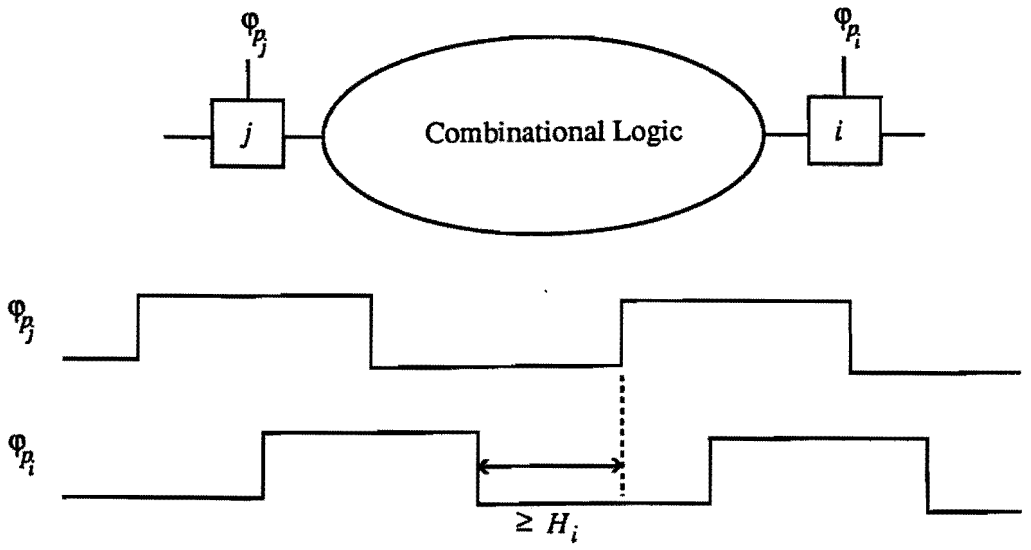


Figure 5: Phase Non-overlap Constraints

by substituting $\delta_j = \delta_{ji} = 0$ and $d_j = T_c - T_{p_j}$ in constraint GC4a and combining the result with the hold time inequality, GC2H, to yield:

$$T_c - T_{p_j} - E_{p_j p_i} \geq a_i \geq H_i \tag{2}$$

This constraint is illustrated in Fig. 5 which clearly shows that the next enabling edge of phase p_j must not occur any sooner than H_i after the latching edge of phase p_i. In other words, this constraint insures that the *end* of the active interval of phase p_i is separated from the *beginning* of the active interval of phase p_j by, at a minimum, the hold time of latch i.

We can now write the system timing constraints exclusively in terms of the late arrival and departure variables, without having to worry about short paths. This is facilitated by introducing one last variable, L_{ij}, defined as the set of synchronizers which are controlled by phase ϕ_j, and which also receive one or more data inputs from synchronizers controlled by phase ϕ_i. Thus, in Fig. 4, $L_{24} = \{9\}$, $L_{32} = \{6, 7, 11\}$, and $L_{41} = \emptyset$ (the empty set). The complete set of restricted system timing constraints, RSTC, is summarized in Table 2. The RSTC are basically the same as the constraints reported in [11]. They differ from these earlier constraints in that they:

- use a different frame of reference for time;
- allow for nonzero hold times;

and,

- use A_i instead of D_i in the setup requirements.

In addition, because the RSTC are derived as a special case of the GSTC, we can still obtain from them the values of the early arrival and departure variables. Specifically,

$$d_i = \begin{cases} T_c - T_{p_i}\,, & i = 1, \cdots, l \quad \text{(latches)} \\ T_c\,, & i = 1, \cdots, l \quad \text{(flip-flops)} \end{cases} \tag{3}$$

and,

$$a_i = \min_j(d_j - E_{p_jp_i}) \quad i,j = 1,\cdots,l \tag{4}$$

5 Calculation of Optimal Cycle Time

The minimum clock cycle time can be found by solving either of the following two optimization problems, depending on which set of system timing constraints is selected:

Program PG: Optimal Cycle Time—GSTC

Minimize T_c
Subject to GSTC (Table 1)

Program PR: Optimal Cycle Time—RSTC

Minimize T_c
Subject to RSTC (Table 2)

These two problems are nonlinear because of the min and max functions in their constraints, implying that their solutions would be difficult to obtain. Fortunately, the nonlinearities in these constraints are mild, and their form suggests an associated, and presumably easier-to-solve, *linear* version of each of the optimization problems. A major result of our work is to show that it is in fact possible to obtain the optimal solutions of PG and PR indirectly by first solving a companion linear program (LP).

We begin by noting that the min and max constraints can be easily *relaxed* and converted to sets of linear inequalities. For example, the late departure synchronization constraint, RC3L, for the ith latch would be replaced by two linear inequalities as follows:

$$D_i = \max(A_i, T_c - T_{p_i}) \Longrightarrow \begin{cases} D_i \geq A_i \\ D_i \geq T_c - T_{p_i} \end{cases}$$

Similarly, the early arrival constraint for the ith synchronizer, GC4a, would be replaced by l linear inequalities:

$$a_i = \min_j(d_j + \delta_j + \delta_{ji} - E_{p_jp_i}) \Longrightarrow a_i \leq (d_j + \delta_j + \delta_{ji} - E_{p_jp_i})$$

Tables 3 and 4 show the results of applying these transformations to the general and restricted system timing constraints shown in Tables 1 and 2. Besides the elimination of the min and max functions, the generation of these *relaxed* constraints involves two additional subtleties:

- Constraint XGC3LDd, $D_i \geq d_i$, is introduced as a substitute for the more obvious choice $D_i \geq T_c - T_{p_i}$, which is now implied because of constraint XGC3LdT, $d_i \geq T_c - T_{p_i}$. The adopted constraint has the advantage of maintaining the chronological order of the early and late signal times (i.e. $d_i \leq D_i$ and $a_i \leq A_i$).

- Redundant bound constraints are eliminated. Thus, only the phase widths T_i and the late departure times D_i are bound from above by T_c.

In what follows we will refer to the *relaxed* versions of the GSTC and RSTC by XGSTC and XRSTC. Using the relaxed constraints, we now define the following linear programs:

Program PXG: Optimal Cycle Time—XGSTC

Minimize T_c
Subject to XGSTC (Table 3)

Program PXR: Optimal Cycle Time—XRSTC

Minimize T_c
Subject to XRSTC (Table 4)

5.1 Minimum Cycle Time—The RSTC Case

We show in this section that the minimum cycle time found by solving the nonlinear program PR is the same as that found by solving the linear program PXR. We also present an algorithm for obtaining the optimal solution of PR by a simple modification of the optimal solution of PXR. These results are essentially the same as those we reported in [11] for an earlier formulation of the system timing constraints.

Denoting the optimal values of PR and PXR by $T_{c,min}^{(R)}$ and $T_{c_{min}}^{(XR)}$, the main result of this section is now simply stated by:

Theorem 5.1 $T_{c,min}^{(R)} = T_{c,min}^{(XR)}$.

Proof: The proof is based on showing that program PR is equivalent to program PXR augmented with extra constraints, and that the optimal value of this augmented linear program is the same as that of program PXR. For the detailed proof steps see [11, Theorem 3.1]. □

This theorem forms the basis for finding the optimal solution of PR by linear programming according to the following algorithm[6]: Several observations should be made about this algorithm:

- The algorithm involves an iterative process which updates the values of the arrival and departure times using the propagation and synchronization constraints of program PR (constraints RC3L and RC4 in Table 2). Throughout this iteration, the values of the clock variables (T_i and e_i for $i = 1, \cdots, k$) are held fixed at the optimal values found by solving program PXR in step (1).

- This iteration is guaranteed to terminate because of the following:

  1. the update equations in step (3) can only cause the arrival and departure times to *decrease*, and,
  2. the departure times are bounded from below by $(T_c - T_{p_i})$

[6] Without loss of generality, we assume for purposes of describing this algorithm that the synchronizers in the circuit are all D-type latches. For circuits containing mixtures of latches and flip-flops, the algorithm should be amended appropriately.

Algorithm 5.1: Find Optimal Solution of Program PR

Comment: m is an iteration counter; g is a convergence flag.

1. *Solve PXR. Denote the optimal values found for the late departure times by D_i^0 for $i = 1, \cdots, l$.*

2. *Set $m = 0$, $g = TRUE$.*

3. *For $i, j = 1, \cdots, l$ evaluate:*

$$A_i^{m+1} = \max_j (D_j^m + \Delta_j + \Delta_{ji} - E_{p_j p_i})$$

$$D_i^{m+1} = \max(A_i^{m+1}, T_c - T_{p_i})$$

4. *If $A_i^{m+1} \neq A_i^m$ or $D_i^{m+1} \neq D_i^m$ for any i, set $g = FALSE$, and increment m.*

5. *If $g == FALSE$ set $g = TRUE$ and go to (3).*

6. *For $i, j = 1, \cdots, l$ set:*

$$d_i = T_c - T_{p_i}$$

$$a_i = \min_j (d_j - E_{p_j p_i})$$

- The update formulas in step (3) have the flavor of a mixed Jacobi/Gauss-Seidel iteration. Other variants are obviously possible. In fact an event-driven update mechanism which only calculates the arrival and departure times which have changed from the previous iteration can be easily implemented. With such an enhancement the cost of the iterative steps can be greatly reduced for large circuits.

- The overall operation of the algorithm can be viewed as a process of "sliding" each of the arrival and departure times to the left (towards the time origin) until any "slack" that may have been introduced by relaxing the corresponding max constraint during step (1) has been reduced to 0 (i.e. until the max constraint is satisfied.)

5.2 Minimum Cycle Time—The GSTC Case

The procedure of finding the minimum cycle time for the GSTC case is complicated by the presence of both min and max functions in the constraints. Thus, unlike the RSTC case, an approach similar to Algorithm 5.1 cannot be guaranteed to yield an optimal solution to program PG under all circumstances. In fact, as we will shortly see, applying such an approach may produce a solution that violates one or more hold time constraints. Fortunately, there is also enough additional information to indicate how this infeasible solution should be modified to make it both feasible and optimal.

Assume, therefore, that we carry out the following two steps:

1. Solve program PXG.

2. Disregarding the hold time constraints, "slide" the solution found in step (1) so that the synchronization and propagation constraints are satisfied.

Let $V_i^H > 0$ denote the amount by which the hold time constraint at synchronizer i is violated at the end of such a procedure; if the hold time constraint is satisfied, then $V_i^H \equiv 0$. The solution obtained by this procedure can, then, be interpreted as that of a linear program—call it PXG′—which is identical to program PXG except that the hold time at each synchronizer is *reduced* by the corresponding violation amount. Specifically, the hold time constraints, $a_i \geq H_i$, are replaced by:

$$a_i \geq H_i - V_i^H \quad i = 1, \cdots, l \tag{5}$$

Furthermore, to account for cases when $H_i < V_i^H$, allow the early arrival times, a_i, to be unrestricted in sign. Finally, let π_i denote the optimal value of the *dual* variable[17] corresponding to the *ith* modified hold time constraint in (5).

With these constructions, it is now possible to obtain the optimal solution of program PG by applying sensitivity analysis techniques of linear programming [17, Sec. 4.4, p. 160] to the solution of program PXG′. In particular, program PG can be "obtained" from program PXG′ by *increasing* the hold time at each synchronizer i by V_i^H. Denoting the optimal values of programs PG and PXG by $T_{c,min}^{(G)}$ and $T_{c,min}^{(XG)}$, we may therefore conclude that:

Conjecture 5.2 $T_{c,min}^{(G)} = T_{c,min}^{(XG)} + \sum_{i=1}^{l} \pi_i V_i^H$.

This result is stated as a conjecture because we have not yet worked out the detailed proof steps. It assumes that the solution of program PXG′ remains basic feasible after the right-hand-side vector is modified.

We close this section by presenting the detailed algorithmic steps for computing the minimum cycle time in the GSTC case: (*** comment on algorithm ***)

6 Example

The timing model introduced in this paper is illustrated by formulating the system timing constraints, both general and restricted, for the example circuit shown in Fig. 6, and solving for the minimum cycle time. The circuit has 8 identical latches—represented by MOS pass transistors in the circuit diagram—with the timing parameters $H_i = 0$ ns, and $S_i = \delta_i = \Delta_i = 10$ ns. The combinational logic stages between latches are labeled C_{ij} where the indices are the numbers denoting an input latch i and an output latch j. Each logic stage is also annotated with the path delay between the respective latches; it is assumed that $\delta_{ij} = \Delta_{ij}$.

The optimal solutions to both problems are given in Table 5. The solutions are also shown graphically in Fig. 7. Each solution consists of a set of signal waveforms over one clock cycle in the global time reference. For clock signals (phases), these waveforms indicate when the signal is enabled (high) and when it is disabled (low). For data signals, the waveforms indicate when the signal is stable and when it is changing. The stable interval for each data signal is further divided into two subintervals to distinguish between the old and new stable values. We adopt the convention that a stable value *becomes old* when it is latched. The solutions in Fig. 7 only list the waveforms of the data signals at latch inputs. The waveforms at latch outputs can be easily obtained by using the latch synchronization constraints[7].

[7] In general, the waveforms of the signals arriving at synchronizer inputs are sufficient to determine the waveforms

For each of the two cases, these results were obtained by solving a linear program (LP) which is derived from the appropriate set of system timing constraints by relaxing the nonlinear min and max constraints (i.e. replacing them with linear inequalities), and then minimizing the cycle time. The optimal solutions were then adjusted manually to satisfy the min and max constraints. Several observations can be made about these results:

- The optimal cycle time found using the GSTC constraints is 20 ns shorter than that found using the RSTC constraints, a performance improvement of 12%. This improvement was possible by partially overlapping the two clock phases.
- The "changing" intervals are wider in the case of the RSTC constraints because of the worst-case assumption of zero minimum propagation delays.
- While not immediately evident from Table 5 or Fig. 7, the dual solutions to the LP provide all the information necessary to identify the critical delays in the circuit. For this circuit, all delays in the left loop are critical, i.e., increasing any one of them causes the optimal cycle time to increase; the delays in the right loop as well as Δ_{45} turn out to be non-critical.
- For comparison purposes, the minimum cycle time when all latches are replaced with flip-flops is 300 ns. Thus, the use of latches improves performance in this case by 45%. The performance gain possible through the use of latches instead of flip-flops is, of course, circuit-dependent.

7 Conclusions and Future Work

The timing model presented in this paper establishes a framework for the study of the dynamic behavior of synchronous digital systems. The model requires that a circuit be decomposed into combinational and sequential parts, a generally difficult task. Furthermore, the dynamic behavior of the sequential structures must be captured in timing "macromodels" similar to those of the D-type latches and flip-flops presented in this paper. Both of these steps, decomposition and macromodeling, require further investigation. Other extensions to the model, for example to handle clock skew, are relatively straightforward and are currently being incorporated.

In parallel with the model development effort, we are investigating efficient algorithms—based on the model—for timing verification and optimal clocking [18]. We are also looking into the derivation of closed-form design criteria for special circuit structures such as pipelines and CPU data paths.

References

[1] N. P. Jouppi, *Timing Verification and Performance Improvement of MOS VLSI Designs*, PhD thesis, Stanford University, Stanford, CA 94305-2192, October 1984.

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

of all other signals in the circuit including signals at nodes which are internal to combinational logic stages.

[3] M. Glesner, J. Schuck, and R. B. Steck, "SCAT- A New Statistical Timing Verifier in a Silicon Compiler System," in *Proceedings of th 23rd Design Automation Conference*, pp. 220–226, 1986.

[4] S. H. Unger and C.-J. Tan, "Clocking Schemes for High-Speed Digital Systems," *IEEE Transactions on Computers*, vol. C-35, no. 10, pp. 880–895, October 1986.

[5] T. G. Szymanski, "LEADOUT : A Static Timing Analyzer for MOS Cicuits," in *ICCAD-86 Digest of Technical Papers*, pp. 130–133, 1986.

[6] J. J. Cherry, "Pearl : A CMOS Timing Analyzer," in *Proceedings of the 25th Design Automation Conference*, pp. 148–153, 1988.

[7] D. E. Wallace and C. H. Sequin, "ATV: An Abstract Timing Verifier," in *Proceedings of the 25th Design Automation Conference*, pp. 154–159, 1988.

[8] N. Ishiura, M. Takahashi, and S. Yajima, "Time-Symbolic Simulation for Accurate Timing Verification of Asynchronous Behavior of Logic Circuits," in *Proceedings of the 26th Design Automation Conference*, pp. 497–502, 1989.

[9] N. Weiner and A. Sangiovanni-Vincentelli, "Timing Analysis in A Logic Synthesis Environment," in *Proceeding of the 26th Design Automation Conference*, pp. 655–661, 1989.

[10] M. R. Dagenais and N. C. Rumin, "On the Calculation of Optimal Clocking Parameters in Synchronous Circuits with Level-Sensitive Latches," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 3, pp. 268–278, March 1989.

[11] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and Design of Latch-Controlled Synchronous Digital Circuits ," in *Proceedings of the 27th Design Automation Conference*, 1990.

[12] T. I. Kirkpatrick and N. R. Clark, "PERT as an Aid to Logic Design," *IBM Journal of Research and Development*, vol. 10, no. 2, pp. 135–141, March 1966.

[13] B. K. Fawcett, *Maximal Clocking Rates for Pipelined Digital Systems*, Master's thesis, University of Illinois, 1975.

[14] T. M. Mcwilliams, "Verification of Timing Constraints on Large Digital Systems," in *Proceedings of the 17th Design Automation Conference*, pp. 139–147, 1980.

[15] V. D. Agrawal, "Synchronous Path Analysis in MOS Circuit Simulator," in *Proceedings of the 19th Design Automation Conference*, pp. 629–635, 1982.

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

[17] D. T. Phillips, A. Ravindran, and J. J. Solberg, *Operations Research: Principles and Practice*, John Wiley and Sons, Inc., 1976.

[18] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Algorithms for Timing Verification and Optimal Clocking of Synchronous Digital Circuits ," Technical report, University of Michigan, Dept of EECS, Ann Arbor, MI 48109-2122, 1990.

RC0. Phase Non-Overlap Constraints:		
	$T_c - T_i - E_{ij} \geq \max_{n \in L_{ij}}(H_n)$	$\forall_{i,j} \ni L_{ij} \neq \emptyset$
RC1. Phase Ordering Constraints:	$e_{i-1} \leq e_i$	$i = 2, \cdots, k$
	$e_k \equiv T_c$	
RC2. Latching (Setup) Constraints:	$A_i \leq T_c - S_i$	$i = 1, \cdots, l$
RC3. Synchronization Constraints:		
RC3L. Latch Synchronization:	$D_i = \max(A_i, T_c - T_{p_i})$	$i = 1, \cdots, l$
RC3F. Flip-Flop Synchronization:	$D_i = T_c$	$i = 1, \cdots, l$
RC4. Combinational (Late Arrival) Propagation Constraints:		
	$A_i = \max_j(D_j + \Delta_j + \Delta_{ji} - E_{p_j p_i})$	$i, j = 1, \cdots, l$
RC5. Bound Constraints:	$0 \leq e_i\ ,\ T_i \leq T_c$	$i = 1, \cdots, k$
	$0 \leq A_i\ ,\ D_i \leq T_c$	$i = 1, \cdots, l$

Table 2: Restricted System Timing Constraints (RSTC)

GC1. Phase Ordering Constraints:	$e_{i-1} \leq e_i$	$i = 2, \cdots, k$
	$e_k \equiv T_c$	
GC2. Latching Constraints:		
GC2H. Hold Time Constraints:	$a_i \geq H_i$	$i = 1, \cdots, l$
GC2S. Setup Time Constraints:	$A_i \leq T_c - S_i$	$i = 1, \cdots, l$
GC3. Synchronization Constraints:		
XGC3L. *Relaxed* Latch Synchronization:		
XGC3Ld. Early Departure:		
XGC3Lda:	$d_i \geq a_i$	$i = 1, \cdots, l$
XGC3LdT:	$d_i \geq T_c - T_{p_i}$	$i = 1, \cdots, l$
XGC3LD. Late Departure:		
XGC3LDA:	$D_i \geq A_i$	$i = 1, \cdots, l$
XGC3LDd:	$D_i \geq d_i$	$i = 1, \cdots, l$
GC3F. Flip-Flop Synchronization:		
GC3Fd. Early Departure:	$d_i = T_c$	$i = 1, \cdots, l$
GC3FD. Late Departure:	$D_i = T_c$	$i = 1, \cdots, l$
XGC4. *Relaxed* Combinational Propagation Constraints:		
XGC4a. Early Arrival:	$a_i \leq (d_j + \delta_j + \delta_{ji} - E_{p_j p_i})$	$i, j = 1, \cdots, l$
XGC4A. Late Arrival:	$A_i \geq (D_j + \Delta_j + \Delta_{ji} - E_{p_i p_i})$	$i, j = 1, \cdots, l$
GC5. Bound Constraints:		
GC5L. Lower Bounds:	$e_i\ ,\ T_i \geq 0$	$i = 1, \cdots, k$
	$a_i\ ,\ A_i\ ,\ d_i\ ,\ D_i \geq 0$	$i = 1, \cdots, l$
GC5U. Upper Bounds:	$T_i \leq T_c$	$i = 1, \cdots, k$
	$D_i \leq T_c$	$i = 1, \cdots, l$

Table 3: XGSTC—Relaxed Version of the GSTC in Table 1

RC0. Phase Non-Overlap Constraints:		
	$T_c - T_i - E_{ij} \geq \max_{n \in L_{ij}}(H_n)$	$\forall_{i,j} \ni L_{ij} \neq \emptyset$
RC1. Phase Ordering Constraints:	$e_{i-1} \leq e_i$	$i = 2, \cdots, k$
	$e_k \equiv T_c$	
RC2. Latching (Setup) Constraints:	$A_i \leq T_c - S_i$	$i = 1, \cdots, l$
RC3. Synchronization Constraints:		
XRC3L. *Relaxed* Latch Synchronization:		
XGC3LDA:	$D_i \geq A_i$	$i = 1, \cdots, l$
XGC3LDT:	$D_i \geq T_c - T_{p_i}$	$i = 1, \cdots, l$
RC3F. Flip-Flop Synchronization:	$D_i = T_c$	$i = 1, \cdots, l$
XRC4. *Relaxed* Combinational (Late Arrival) Propagation Constraints:		
	$A_i \geq (D_j + \Delta_j + \Delta_{ji} - E_{p_j p_i})$	$i, j = 1, \cdots, l$
RC5. Bound Constraints:		
RC5L. Lower Bounds:	$e_i\ ,\ T_i \geq 0$	$i = 1, \cdots, k$
	$A_i\ ,\ D_i \geq 0$	$i = 1, \cdots, l$
RC5U. Upper Bounds:	$T_i \leq T_c$	$i = 1, \cdots, k$
	$D_i \leq T_c$	$i = 1, \cdots, l$

Table 4: XRSTC—Relaxed Version of the RSTC in Table 2

	GSTC				*RSTC*			
Clock	T_1	e_1	T_2	e_2	T_1	e_1	T_2	e_2
Phases	95	100	105	165	75	100	85	185
Latch #	a_i	A_i	d_i	D_i	a_i	A_i	d_i	D_i
1	105	105	105	105	0	125	110	125
2	60	60	60	60	25	60	100	100
3	70	70	70	70	0	110	110	110
4	155	155	155	155	25	175	100	175
5	125	155	125	155	0	175	110	175
6	90	120	90	120	25	120	100	120
7	20	50	70	70	0	50	110	110
8	95	95	95	95	25	115	100	115

Table 5: Optimal Solutions for the Example Circuit

Algorithm 5.2: Find Minimum Cycle Time for Program PG

Comment: m is an iteration counter; g is a convergence flag; t is a temporary time variable

1. *Solve PXG. Denote the optimal values found for the early and late departure times by d_i^0 and D_i^0 for $i = 1, \cdots, l$. Denote the optimal cycle time by $T_{c,min}^{(XG)}$.*

2. *Set $m = 0$, $g = TRUE$.*

3. *For $i, j = 1, \cdots, l$ evaluate:*

   (a) *Late arrival and departure times:*

$$A_i^{m+1} = \max_j (D_j^m + \Delta_j + \Delta_{ji} - E_{p_j p_i})$$

$$D_i^{m+1} = \max(A_i^{m+1}, T_c - T_{p_i})$$

   (b) *Early arrival and departure times:*

$$t = \min_j (d_j^m + \delta_j + \delta_{ji} - E_{p_j p_i})$$

$$V_i^H = \max(0, H_i - t)$$

$$a_i^{m+1} = \max(t, H_i)$$

$$d_i^{m+1} = \max(a_i^{m+1}, T_c - T_{p_i})$$

4. *If $A_i^{m+1} \neq A_i^m$ or $D_i^{m+1} \neq D_i^m$ or $a_i^{m+1} \neq a_i^m$ or $d_i^{m+1} \neq d_i^m$ for any i, set $g = FALSE$, and increment m.*

5. *If $g == FALSE$ set $g = TRUE$ and go to (3).*

6. *Compute the minimum cycle time from:*

$$T_{c,min}^{(G)} = T_{c,min}^{(XG)} + \sum_{i=1}^{l} \pi_i V_i^H$$

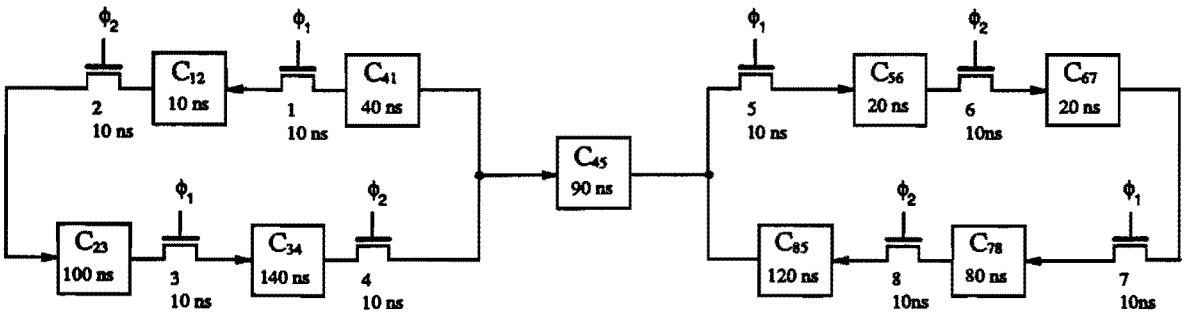
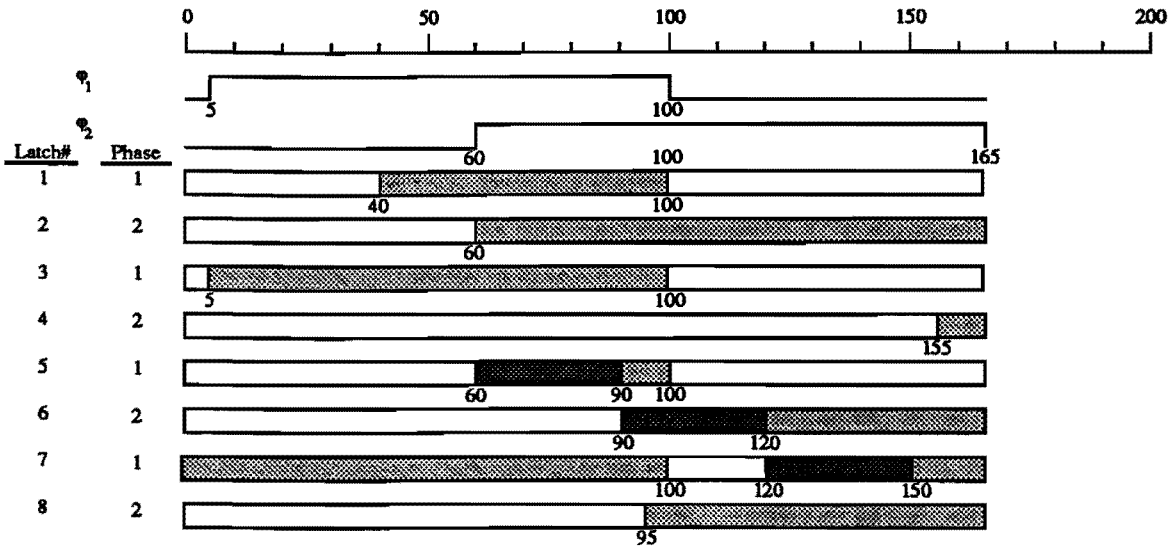
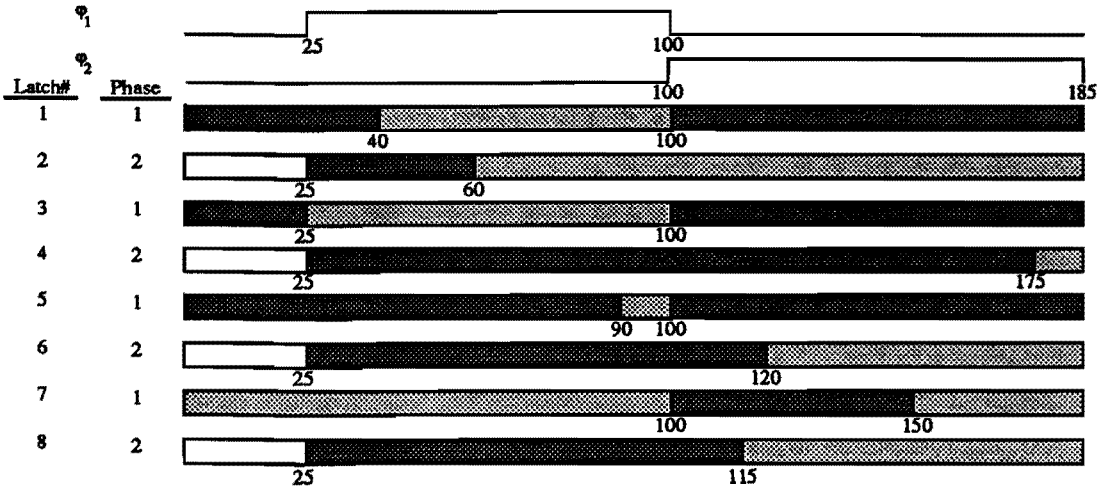


Figure 6: Example Circuit



(a) Solution obtained using the general system timing constraints (GSTC)



(b) Solution obtained using the restricted system timing constraints (RSTC)

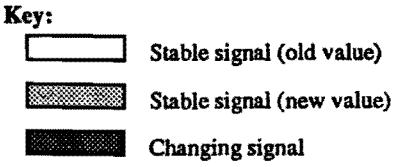


Figure 7: Timing of Arriving Signals for the Example Circuit