Synchronization of Pipelines

by

Karem A. Sakallah, Trevor N. Mudge, Timothy M. Burks, and Edward S. Davidson

CSE-TR-97-91

Computer Science and Engineering Division
Room 3402 EECS Building

THE UNIVERSITY OF MICHIGAN

Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Synchronization of Pipelines

Karem A. Sakallah Trevor N. Mudge Timothy M. Burks Edward S. Davidson
Advanced Computer Architecture Lab
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

Phone: 313-936-1350
FAX: 313-763-4617
E-mail: karem@eecs.umich.edu

February 11, 1991

Abstract

This paper presents a new timing model of pipelines and uses it to determine the minimum cycle time in terms of stage delays. The model accounts for short- and long-path delays, the use of both level-sensitive latches and edge-triggered flip-flops as synchronizing elements, and *wave pipelined* operation. We give results for single-phase clocking and for a restricted form of multi-phase clocking, and show that the region of feasible solutions for the former maybe non-convex. We also describe a program, *pipe*T_c , which generates these optimal solutions from pipeline specifications and illustrate its application on some examples.

List of Symbols

i , j : Indices used to identify pipeline stages/synchronizers.

p , r : Indices used to identify clock phases.

p_i : Index of clock phase used to control synchronizer i.

a_i , A_i : Early and late signal arrival times at stage i.

d_i , D_i : Early and late signal departure times from stage i.

δ_i , Δ_i : Minimum and maximum propagation delays from synchronizer $i-1$ to synchronizer i.

C : Concurrency in the pipeline.

e_p : Time, in global frame-of-reference, at which clock phase p ends (i.e. when its latching edge occurs).

E_{pr} : Phase shift from clock phase p to clock phase r.

$\mathcal{E}_{i-1,i}$: Phase shift from stage $i-1$ to stage i.

H_i : Hold time of synchronizer i.

k : Number of clock phases.

m : Width (in bits) of the pipeline datapath.

n : Number of pipeline stages.

ν_i : Degree of wave pipelining in stage i.

S_i : Setup time of synchronizer i.

T_c : Clock cycle time.

T_p : Width of active interval of phase p.

U : Utilization of the pipeline.

ϕ_p : Name of clock phase whose index is p.

w : Minimum allowable clock pulse width.

1 Introduction

Pipelining is frequently used to speed up the execution of a sequence of computations by dividing each into n consecutive subcomputations and overlapping their execution. Theoretically, this should yield a factor of n performance improvement over the non-pipelined

case. This maximum is rarely achieved, however, because of dependencies among the operations and overhead due to clocking [1]. Performance can be defined as the sustained number of operations per unit time, and can be expressed as:

$$MOPS = \frac{U(n) \times 10^3}{T_c(n)}$$

where *MOPS* stands for millions of operations per second, $0 \leq U(n) \leq 1$ is the *utilization* of the n-stage pipeline, and $T_c(n)$ is the clock cycle time, in nanoseconds, at each pipe stage. Typically, $U(n)$ is a decreasing function of n which is determined empirically through simulations or benchmarking. $T_c(n)$ is also a decreasing function of n, in general, but it also depends on circuit delays and clocking parameters. Optimal pipeline design seeks to find the value of n which maximizes *MOPS*. This is usually done in two steps: 1) Determining $U(n)$ for a suitable range of n by analyzing the operation inter-dependencies for an appropriate set of benchmark computations. This is a purely "architectural" analysis which disregards all implementation details. 2) Determining the minimum $T_c(n)$ for the same range of n. Generally, this is a synthesis problem which involves examining the logic design of various pipelines, and finding those which yield the minimum cycle times. This paper addresses one aspect of the second step, namely, determining the minimum cycle time, $T_{c,\min}$, for an n-stage pipeline in terms of circuit delays. The problem has been addressed previously by a number of authors including [2, 3, 4, 1]. This previous work dealt mostly with simple clocking paradigms. Furthermore, the analysis was typically based on examination of a single pipe stage. In contrast, in this paper we propose a timing model of pipelines that accounts for more complex clocking and for the temporal interactions among the various pipe stages.

The remainder of this paper is organized as follows. Our pipeline model is developed in Sec. 2. In Sec. 3 we derive the minimum cycle time for single-phase and a restricted-form of multi-phase clocking. Sec. 4 describes a program, *pipe*T_c , which produces the corresponding optimal clock schedules from a specification of pipeline parameters. We use the program on two pipeline examples to point out some of the (non-obvious) features of the optimal solutions. Conclusions and suggestions for future work are summarized in Sec. 5.

2 Pipeline Model

Our pipeline model is shown in Fig. 1. Unlike earlier formulations, such as those given in [2, 3, 4, 1], the pipe stages in our model form a simple closed loop. This is a more realistic configuration than an open-ended pipeline because it accounts for the the timing of the source and sink of the data flowing in the pipeline. For example, one or more stages in such a pipeline can be used to model the "memory" used to supply operands for the computation, and to receive results from it. Furthermore, the analysis of closed pipelines can be easily adapted to model open-ended pipelines. In particular, an open-ended pipe can be transformed into a closed pipe by adding an artificial stage whose timing parameters reflect the input/output signal timing specifications of the original open pipe.

The pipe stages are numbered consecutively from 0 to $(n-1)$. The datapath through the pipeline is assumed to be m bits wide, $m \geq 1$. Each pipe stage consists of a bank of m synchronizing elements (level-sensitive latches or edge-triggered flip-flops) followed by

combinational circuitry. Data flow through the pipeline is regulated by a k-phase clock, where $1 \leq k \leq n$. Stage i is characterized by the following parameters:

p_i : an integer denoting the clock phase used to control the synchronizer at the output of stage i (henceforth referred to as synchronizer i).

S_i : non-negative setup time of synchronizer i relative to latching edge of phase p_i.

H_i : non-negative hold time of synchronizer i relative to latching edge of phase p_i.

δ_i , Δ_i : minimum and maximum propagation delays ($0 \leq \delta_i \leq \Delta_i$) from the input of synchronizer $i-1$ to the input of synchronizer i.
Note that this definition of stage delay lumps together the two components of signal delay, namely the synchronizer delay and the combinational logic delay.

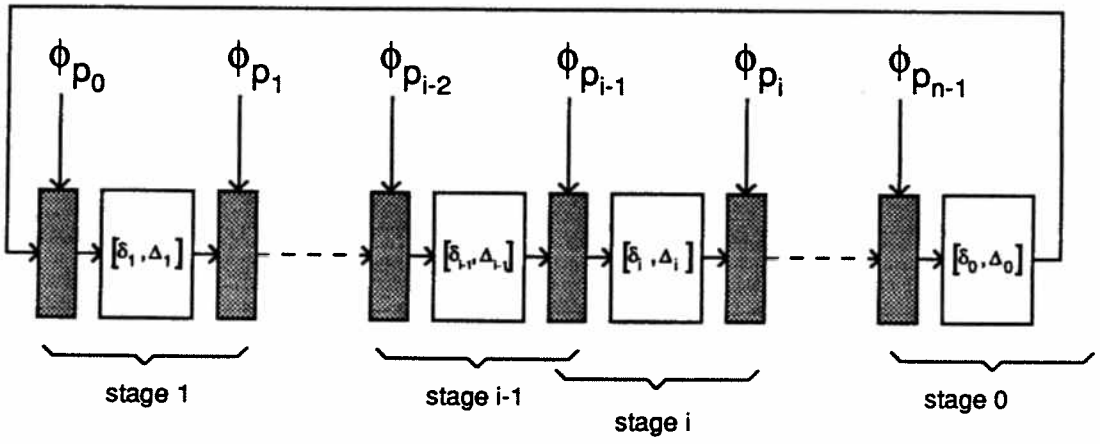


Figure 1: n-stage pipeline. Shaded boxes represent the synchronizers

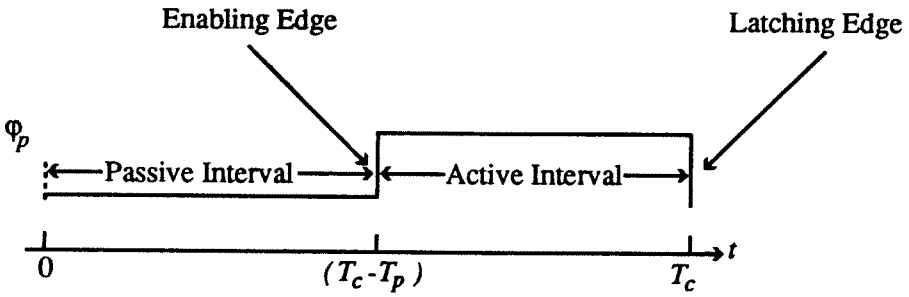


Figure 2: Clock phase ϕ_p and its local time zone

We base the steady-state behavior of such pipelines on the general model of synchronous

operation introduced in [5]. The salient features of this model, as they relate to the pipeline, are summarized below. In addition, we extend the model to allow for *wave pipelined* [6] operation, and examine the relationships among concurrency in the pipeline, clocking, and the degree of wave pipelining.

2.1 Clocking Model

The clocking model is described in terms of a *temporal* rather than a *logical* framework based on the concept of *periodic phases* which define *local time zones* related by *phase shift operators*. In this model, a k-phase clock is considered to be a collection of k periodic signals ϕ_1 , ϕ_2 , $\cdots$, ϕ_k — referred to as the *phases* — with a common cycle time T_c. Each phase ϕ_p divides the clock cycle into two intervals: an *active* interval of duration T_p, and a *passive* interval of duration $(T_c - T_p)$. 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 are 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. Associated with the phase is a *local time zone*, as shown in Fig. 2, such that the passive interval of the phase starts at $t = 0$, its enabling edge occurs at $t = T_c - T_p$, and its latching edge occurs at $t = T_c$. The temporal relationships among the k phases (i.e. among the different time zones) are established by an arbitrary choice of a *global time reference*. We introduce e_p to denote the time, relative to this global time reference, at which phase ϕ_p *ends* (i.e. when its latching edge occurs). Finally, we define a *phase shift* operator:

$$E_{pr} \equiv \begin{cases} (e_r - e_p)\,, & e_r > e_p \\ (T_c + e_r - e_p)\,, & e_r \leq e_p \end{cases} \tag{1}$$

which takes on positive values in the range $(0, T_c]$ (see Fig. 3). When subtracted from a time variable in the *current* local time zone of ϕ_p, E_{pr} changes the frame of reference to the *next* local time zone of ϕ_r, taking into account a possible cycle boundary crossing.

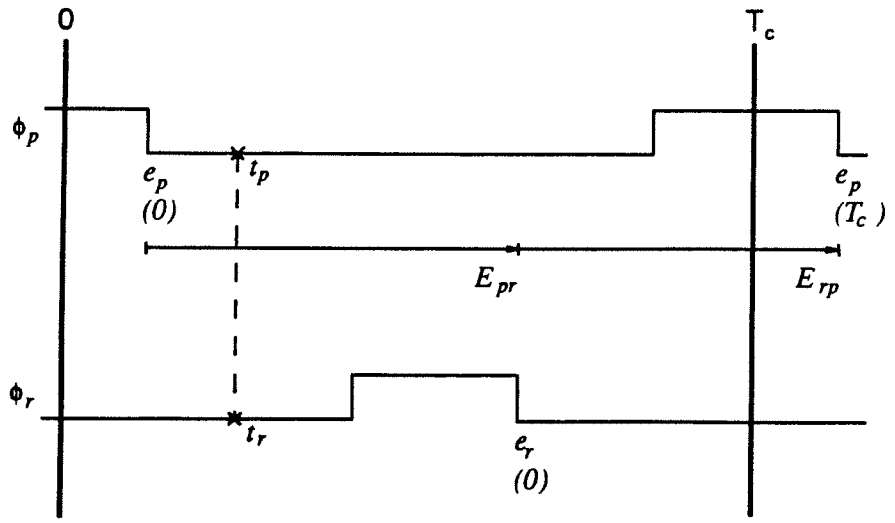
2.2 Timing Constraints

For timing purposes, it is sufficient to characterize a data signal with respect to 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 synchronizer input of pipeline stage 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 input of 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 timing model of the pipeline can now be expressed by the following constraints and equations [5] for $i = 0, \cdots, n-1$:

Clock Constraints which express limitations on clock generation and distribution. This set should at least include the following minimum pulse width constraints:

$$T_{p_i} \geq w \tag{2}$$

$$T_c - T_{p_i} \geq w \tag{3}$$



$$t_r = t_p - E_{pr}$$

Figure 3: Interpretation of the phase shift operator

where w is a specified parameter. In addition, to simplify the design of the clock generator we may include "regularity" constraints such as:

$$T_1 = T_2 = \cdots = T_k \tag{4}$$

It is important to point out that the phase signals are *not* required to be non-overlapping.

Latching Constraints which express the conditions necessary for capturing valid data values at each of the synchronizers. They consist of two sets of requirements which, together, insure that the data signal at the input of a synchronizer is stable for a sufficient period of time before and after the occurrence of the latching edge of the corresponding clock. Mathematically,

$$a_i \geq H_i \tag{5}$$

$$A_i \leq T_c - S_i \tag{6}$$

Synchronization Equations which *macromodel* the temporal behavior of different types of synchronizing elements. Specifically, for D-type synchronizers, they express the departure times of the output data signals in terms of the arrival times of the corresponding input data signals as well as those of the enabling and latching clock edges. Thus, for latches, we obtain:

$$d_i = \max(a_i, T_c - T_{p_i}) \tag{7}$$

$$D_i = \max(A_i, T_c - T_{p_i}) \tag{8}$$

For flip-flops, the macromodel is simpler:

$$d_i = T_c \tag{9}$$

$$D_i = T_c \tag{10}$$

Propagation Equations which model the delay of the combinational stages in the pipeline, including the propagation through the input synchronizer. They express the arrival times of data at stage i in terms of the corresponding departure times from stage $(i-1) \bmod n$, taking into account the change in the frame-of-reference from phase p_{i-1} to phase p_i:[1]

$$a_i = d_{i-1} + \delta_i - \mathcal{E}_{i-1,i} \tag{11}$$

$$A_i = D_{i-1} + \Delta_i - \mathcal{E}_{i-1,i} \tag{12}$$

where $\mathcal{E}_{i-1,i}$ is the amount of phase shift *from stage* $i-1$ *to stage* i. In [5], this was defined to be equal to the phase shift *from clock phase* p_{i-1} *to clock phase* p_i, i.e. $\mathcal{E}_{i-1,i} \equiv E_{p_{i-1}p_i}$. This definition limited signal propagation to consecutive cycles of phases p_{i-1} and p_i, i.e. signals launched from stage $i-1$ in any given cycle of phase p_{i-1} had to arrive and be correctly latched at stage i by the immediately following cycle of phase p_i. We extend this definition here to allow for signal propagation over multiple clock cycles by introducing the parameter ν_i to indicate the number of *additional* clock cycles available for signals to propagate from stage $i-1$ to stage i. Thus,

$$\mathcal{E}_{i-1,i} \equiv E_{p_{i-1}p_i} + \nu_i T_c \tag{13}$$

Note that the addition of an integer number of clock cycles to the clock phase shift has the effect of changing the frame-of-reference from the current local time zone of phase p_{i-1} to that of phase p_i, $(1+\nu_i)$ cycles in the future. In particular, for $\nu_i = 0$ it reverts to the earlier definition which limits phase shifts to consecutive cycles. It is also convenient to view these phase shifts as *negative delays* that effectively reduce the stage propagation delays.

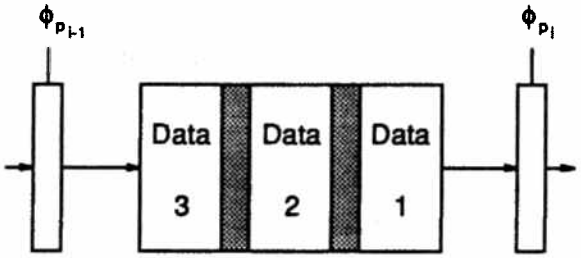


Figure 4: Wave Pipelining

2.3 Operation Modes

Allowing for signals to propagate across multiple clock cycles has the potential of reducing the cycle time below what is possible with single-cycle propagation. However, for such operation to be feasible the minimum combinational delays must be sufficiently large to maintain adequate temporal separation between consecutive *waves* of signals (see Fig. 4).

[1] Index arithmetic in what follows will always be modulo n. To keep the equations from becoming too cluttered, the mod operator will be dropped and assumed to be implied.

Reliance on logic delay, rather than synchronizing elements, to prevent interference between consecutive data waves has been dubbed *wave pipelining* in [6]. This phenomenon will occur in any pipe stage for which $\nu_i > 0$. Furthermore, as we show later, it may also occur for latch-controlled pipes even when $\nu_i = 0$. In fact, wave pipelining occurs, for some portion of the clock cycle, in any pipe stage whose delay exceeds the clock cycle time. We will refer to ν_i as the *degree of wave pipelining*, with the understanding that some limited form of wave pipelining may exist in latch-controlled pipes with $\nu_i = 0$.

The number of simultaneous operations in an n-stage pipeline need not be equal to n. Depending on the nature of the clocking scheme, the differences between the minimum and maximum delays in each stage, and the spread of the maximum delays across all stages, it may be possible to operate the pipeline so that the number of simultaneous operations in progress at any given time is less than or greater than n. We capture this notion by introducing C, the *concurrency* in the pipeline, which can be easily related to the clock phase shifts and wave pipelining parameters by:

$$C = \frac{1}{T_c} \sum_{i=0}^{n-1} E_{p_{i-1}p_i} + \sum_{i=0}^{n-1} \nu_i \tag{14}$$

C can be thought of as the number of *virtual* pipeline stages, and may also be interpreted as the number of data wavefronts in the pipeline at any given time. As Fig. 5 shows, a particular level of concurrency may be achieved by a variety of combinations of clocking schemes and wave pipelining. For example, a concurrency of 4 in a 4-stage pipeline may be obtained by a 4-phase clock where each pipe stage is allocated a fraction of the clock cycle such that $\sum E_{p_{i-1}p_i} = T_c$, and $\sum \nu_i = 3$. Alternatively, $\sum E_{p_{i-1}p_i} = 2T_c$, and $\sum \nu_i = 2$.

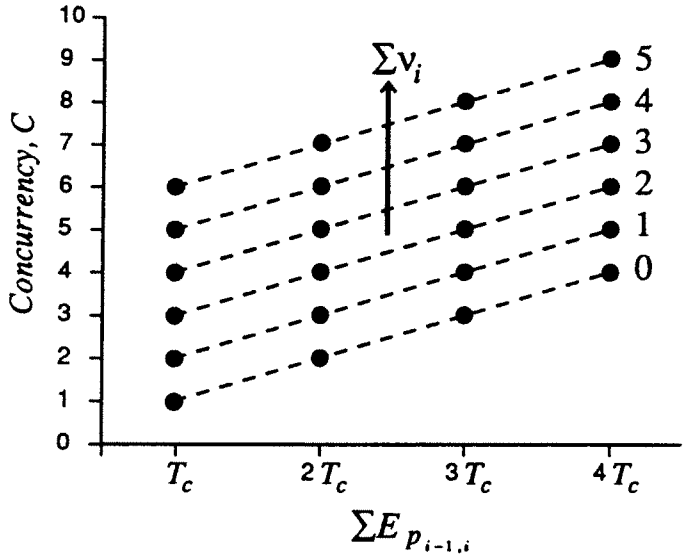


Figure 5: Concurrency as a function of clock phase shifts and degree of wave pipelining for a 4-stage pipeline

We limit our attention in this paper to those clocking schemes which maximize C for a given level of wave pipelining, namely those for which the sum of the clock phase shifts

around the pipeline stages is equal to nT_c. Recalling that each of the individual phase shifts is at most one clock cycle, this restriction implies that $E_{p_{i-1}p_i} = T_c$ for each of the n stages. Clocking schemes for which this restriction applies include single-phase clocks and the restricted form of multi-phase clocking shown in Fig. 6, which will be referred to as *coincident* multi-phase clocking since the latching edges of all k phases *coincide* in time[2]. We further restrict the scope of the investigation by assuming that ν_i is the same for all stages. Thus, we let $\nu_i = \nu$.

With these restriction, the concurrency C becomes:

$$C = (1 + \nu)n \tag{15}$$

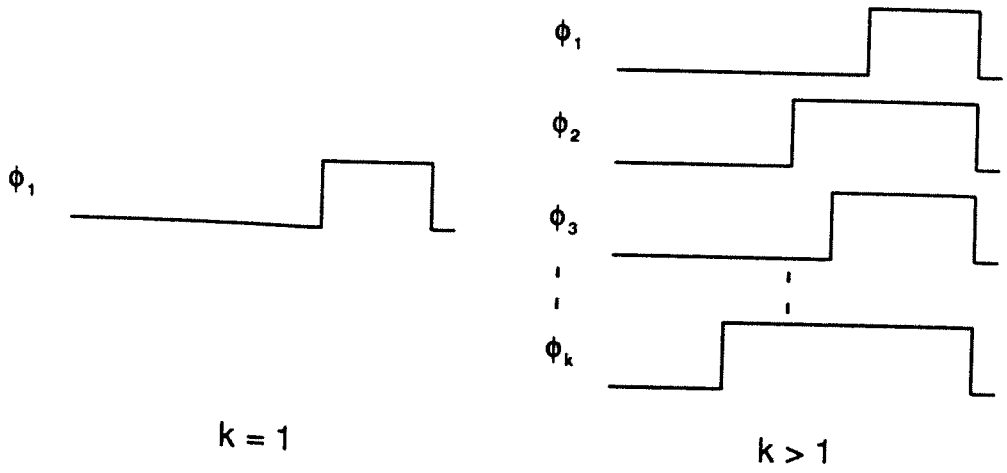


Figure 6: Clocks with maximum possible phase shift between phases

2.4 Summary

In the derivations to follow, it will be convenient to view this model as consisting of *three* distinct sets of constraints:

- **Pulse-Width Constraints** expressed by (2) and (3).
- **Long-Path (*Late-Signal*) Constraints** involving the late arrival and departure times and expressed by the setup inequalities (6), the propagation equations (12), and the synchronization equations (10) or (8).
- **Short-Path (*Early-Signal*) Constraints** involving the early arrival and departure times and expressed by the hold inequalities (5), the propagation equations (11), and the synchronization equations (9) or (7).

[2]For general multi-phase clocks, the existence of fractional phase shifts (i.e. phase shifts smaller than a full cycle) limits the maximum number of clock cycles to $(n-1)$.

3 Optimal Cycle Time Calculation

Within the limitations specified above, i.e. $E_{p_{i-1}p_i} = T_c$ and $\nu_i = \nu$, we derive in this section the minimum cycle time for three different cases:

1. Pipelines controlled by edge-triggered flip-flops.
2. Pipelines controlled by latches and a single-phase clock.
3. Pipelines controlled by latches and a coincident n-phase clock.

In each case the derivation starts by finding expressions for the early and late arrival times at stage i in terms of the clock variables and circuit delays. These expressions are then combined with the hold and setup requirements to obtain the short- and long-path constraints.

3.1 Flip-Flops

Arrival Times: The early and late arrival times at stage i are obtained by substituting the flip-flop synchronization equations (9) and (10) into the propagation equations (11) and (12):

$$a_i = T_c + \delta_i - (1+\nu)T_c = \delta_i - \nu T_c \tag{16}$$

and,

$$A_i = T_c + \Delta_i - (1+\nu)T_c = \Delta_i - \nu T_c \tag{17}$$

Long-Path Constraints: Combining (17) with the setup inequality (6) yields:

$$(1+\nu)T_c \geq \Delta_i + S_i \tag{18}$$

Short-Path Constraints: Combining (16) with the hold inequality (5) yields:

$$\nu T_c \leq \delta_i - H_i \tag{19}$$

Solution: Examination of (19) and (18) reveals that a solution exists under the following conditions:

$$\begin{aligned} \nu = 0 \quad &: \quad \delta_i \geq H_i \quad \text{for } i = 0, \cdots, n-1 \\ \nu \geq 1 \quad &: \quad \min_i \left(\frac{\delta_i - H_i}{\nu} \right) \geq \max_i \left(\frac{\Delta_i + S_i}{1+\nu} \right) \end{aligned}$$

and the minimum cycle time is:

$$T_{c,\min} = \max_i \left(\frac{\Delta_i + S_i}{1+\nu} \right) \tag{20}$$

The phase widths T_{p_i} can be chosen arbitrarily as long as they satisfy the minimum pulse width constraints (2) and (3):

$$w \leq T_{p_i} \leq T_{c,\min} - w \tag{21}$$

Note that $T_{c,\min}$ in (20) is independent of the phase widths because of edge-triggering. Hence this solution is equally applicable to single- as well as to coincident multi-phase clocking. Since multi-phase clocking offers no particular advantage in this case, the regularity constraints (4) can be used to obtain a single-phase solution that satisfies (21). In contrast, when latches are used as synchronizers, single- and multi-phase clocking may yield different optima. We discuss these cases next.

3.2 Latches—Single-Phase Clock

Arrival Times: The solution in the case of latches is considerably more complicated because of the coupling between signal arrival and departure times through the latch synchronization equations (7) and (8). Unlike the flip-flop case, obtaining expressions for the arrival times at stage i requires the substitution of the synchronization and propagation equations of *all* pipe stages. Thus, the early arrival time at stage i is calculated as follows (recall that for single-phase clocking $p_i = 1$ and $T_{p_i} = T_1$ for all i):

$$\begin{aligned}
a_i &= d_{i-1} + \delta_i - (1+\nu)T_c \\
&= \max(a_{i-1}, T_c - T_1) + \delta_i - (1+\nu)T_c \\
&= \max(a_{i-1} + \delta_i - (1+\nu)T_c, \delta_i - T_1 - \nu T_c) \\
&= \max(d_{i-2} + \delta_{i-1} + \delta_i - (2+2\nu)T_c, \delta_i - T_1 - \nu T_c) \\
&= \max(\max(a_{i-2}, T_c - T_1) + \delta_{i-1} + \delta_i - (2+2\nu)T_c, \delta_i - T_1 - \nu T_c) \\
&= \max(a_{i-2} + \delta_{i-1} + \delta_i - (2+2\nu)T_c, \delta_{i-1} + \delta_i - T_1 - (1+2\nu)T_c, \delta_i - T_1 - \nu T_c) \\
&= \cdots \\
&= \max(a_i + \delta_{i-n} + \cdots + \delta_i - (n+n\nu)T_c, \\
&\qquad \delta_{i-n+1} + \cdots + \delta_i - T_1 - (n-1+n\nu)T_c, \cdots, \\
&\qquad \delta_{i-1} + \delta_i - T_1 - (1+2\nu)T_c, \delta_i - T_1 - \nu T_c)
\end{aligned}$$

which can be expressed more conveniently as:

$$\begin{aligned}
a_i = \max\Bigg\{ & a_i + \left(\sum_{j=i-n}^{i} \delta_j\right) - (n+n\nu)T_c, \\
& \max_{0 \le l \le (n-1)} \left[\left(\sum_{j=i-l}^{i} \delta_j\right) - T_1 - (l+\nu+l\nu)T_c \right] \Bigg\}
\end{aligned} \tag{22}$$

Similarly, the late arrival time at stage i is:

$$\begin{aligned}
A_i = \max\Bigg\{ & A_i + \left(\sum_{j=i-n}^{i} \Delta_j\right) - (n+n\nu)T_c, \\
& \max_{0 \le l \le (n-1)} \left[\left(\sum_{j=i-l}^{i} \Delta_j\right) - T_1 - (l+\nu+l\nu)T_c \right] \Bigg\}
\end{aligned} \tag{23}$$

Note that the max functions in these expressions involve $n+1$ arguments in which, except for the first argument, the only variables are the two clock variables T_c and T_1.

Long-Path Constraints: Expression (23) implies the following $n+1$ inequalities:

$$A_i \ge A_i + \left(\sum_{j=i-n}^{i} \Delta_j\right) - (n+n\nu)T_c \tag{24}$$

$$A_i \ge \left[\left(\sum_{j=i-l}^{i} \Delta_j\right) - T_1 - (l+\nu+l\nu)T_c \right] \quad l = 0, \cdots, n-1 \tag{25}$$

Eliminating A_i from the first inequality, we immediately obtain the following lower bound on T_c:

$$T_c \geq \frac{1}{n(1+\nu)} \sum_{j=i-n}^{i} \Delta_j = \frac{1}{n(1+\nu)} \sum_{j=0}^{n-1} \Delta_j \equiv \frac{\overline{\Delta}}{1+\nu} \tag{26}$$

which confirms the intuitive fact that the cycle time cannot be less than the average pipeline stage delay when $\nu = 0$. Combining each of the remaining n inequalities with the setup constraint (6) we eliminate A_i to obtain:

$$(1+l)(1+\nu)T_c + T_1 \geq \left(\sum_{j=i-l}^{i} \Delta_j \right) + S_i \quad l = 0, \cdots, n-1 \tag{27}$$

While the physical interpretation of each of these inequalities is not as obvious as that of (26), it is still rather simple: the time available for a signal to propagate down the $(l+1)$ pipe stages ending at stage i, and to be correctly latched at stage i, is $(1+l)(1+\nu)$ clock cycles plus the phase width T_1 which represents the "extra" time due to the use of level-sensitive latches. Since each of these inequalities must be true for all n pipe stages, we finally obtain:

$$(1+l)(1+\nu)T_c + T_1 \geq \max_{0 \leq i \leq (n-1)} \left[\left(\sum_{j=i-l}^{i} \Delta_j \right) + S_i \right] \quad l = 0, \cdots, n-1 \tag{28}$$

Thus the long-path constraints have been reduced to the $n+1$ inequalities in (26) and (28) which together define a convex set in the T_c/T_1 solution space as shown in Fig. 7.

Short-Path Constraints: Proceeding as we did for the late arrival time at stage i, we obtain the following inequalities that must be satisfied by the early arrival time:

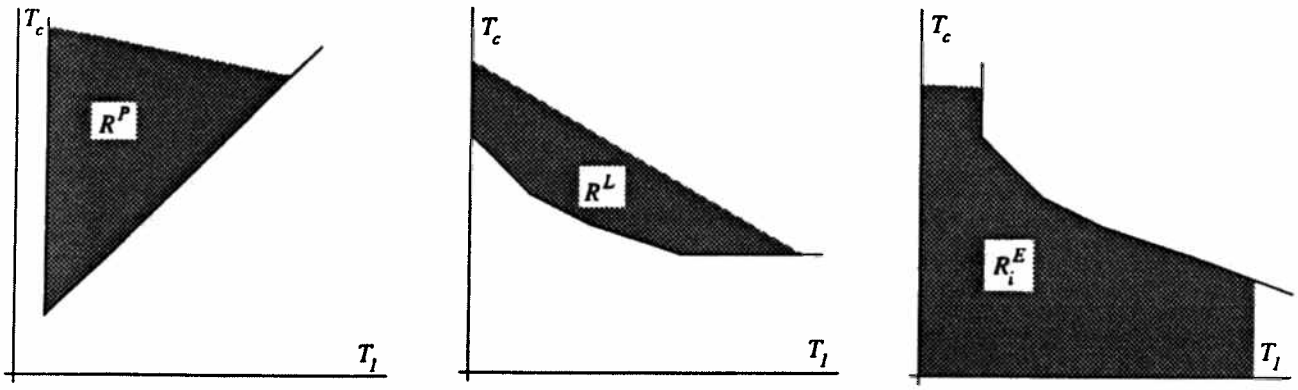
$$a_i \geq a_i + \left(\sum_{j=i-n}^{i} \delta_j \right) - (n + n\nu)T_c \tag{29}$$

$$a_i \geq \left[\left(\sum_{j=i-l}^{i} \delta_j \right) - T_1 - (l + \nu + l\nu)T_c \right] \quad l = 0, \cdots, n-1 \tag{30}$$

The first of these is redundant since it is subsumed by the corresponding max-delay inequality (24). The remaining n inequalities may now be combined with the hold requirement (5) to eliminate a_i and arrive at the set of short-path constraints:

$$(l + \nu + l\nu)T_c + T_1 \leq \left(\sum_{j=i-l}^{i} \delta_j \right) - H_i \qquad \text{for } \textit{at least} \text{ one } l \in \{0, \cdots, n-1\} \tag{31}$$

Note that, unlike the corresponding long-path inequalities (27) which must *all* be satisfied, the above set of n short-path inequalities is satisfied if *at least one* of them is satisfied. In other words, the feasible region defined by the set of n inequalities in (27) is the *intersection* of n separate regions, whereas that defined by the inequalities in (31) is the *union*



(a) Pulse-Width Constraints (b) Long-Path Constraints (c) Short-Path Constraints

Figure 7: Single-Phase Feasible Regions

of n separate regions. This in turn implies that while the region defined by (27) is guaranteed to be convex, that defined by (31) is guaranteed to be non-convex as shown in Fig. 7.

Solution: Denoting the feasible regions corresponding to the pulse-width and long-path (late-signal) constraints by R^P, R^L and that corresponding to the short-path (early-signal) constraints of stage i by R_i^E, the overall region of feasibility can be expressed as:

$$R = R^P \bigcap R^L \bigcap R_0^E \bigcap \cdots \bigcap R_{n-1}^E \tag{32}$$

Due to the non-convexity of R_i^E, R may be non-convex or even disconnected. Examples of these cases will be illustrated in Sec. 4. In any case, assuming that $R \neq \emptyset$, at the optimal solution one or more of the long-path constraints (26) and (28) must be active (satisfied as an equation). This observation forms the basis for a directed-search algorithm to find the minimum cycle time. Basically, the search begins by finding the smallest possible cycle time that satisfies the minimum pulse-width and long-path constraints ($R^P \bigcap R^L$). This point corresponds to the intersection of $T_c - T_1 = w$ and one of the $n+1$ long-path constraints. This solution is now examined to see if it satisfies all of the short-path constraints. If it does, then it is optimal, otherwise we "climb" up the lower periphery of R^L until we either satisfy all the short-path constraints, or we reach the other end of the minimum pulse width region ($T_1 = w$) without satisfying all the short-path constraints. If the latter obtains, the problem is infeasible.

A simpler solution procedure results if the short-path constraints are restricted so that (31) holds for $l = 0$. This leads to a much simpler requirement

$$\nu T_c + T_1 \leq \min_{0 \leq i \leq (n-1)} (\delta_i - H_i) \tag{33}$$

that corresponds to a convex region. When this *conservative* short-path constraint is combined with the long-path constraints (26) and (28), we obtain the following expression for

the minimum cycle time:

$$T_{c,\min} = \max\left\{\frac{\overline{\Delta}}{1+\nu}, \max_{l} \frac{\max_i\left[\left(\sum_{j=i-l}^{i}\Delta_j\right)+S_i\right] - \min_i(\delta_i - H_i)}{1+l+l\nu}\right\} \tag{34}$$

Feasibility of this solution is determined by combining (33) with the minimum pulse-width requirement (2):

$$\nu T_c \leq \min_i(\delta_i - H_i) - w \tag{35}$$

from which we conclude that the above solution is feasible if:

$$\nu = 0 \quad : \quad \min_i(\delta_i - H_i) \geq w$$
$$\nu \geq 1 \quad : \quad \left(\frac{\min_i(\delta_i - H_i) - w}{\nu}\right) \geq T_{c,\min}$$

We will refer to (34) as the *conservative* single-phase solution.

3.3 Latches—Coincident Multi-phase Clocks

Allowing the phase widths to be independently-adjustable at each of the n pipe stages converts the single-phase clock into a coincident n-phase clock. In this case, $p_i = i$, and the clock is specified by $n+1$ variables: $T_c, T_0, \cdots, T_{n-1}$. An n-phase clock may lead to a lower cycle time than a single-phase clock.

Arrival Times: The flexibility to adjust the phase widths independently allows the synchronization equations (7) and (8) to be simplified to $d_i = D_i = T_c - T_i$. This can be justified as follows:

> *Suppose that $D_i > T_c - T_i$ for some stage i at the optimal solution. Then $D_i = A_i > T_c - T_i$. Since changing T_i can only directly affect the departure times from stage i, it should be obvious that T_i can be decreased until $D_i = A_i = T_c - T_i$ without affecting the optimal cycle time. Note also that decreasing T_i increases the margin by which the hold requirement is satisfied at stage $i+1$.*

Under these conditions, the arrival times at stage i become:

$$a_i = T_c - T_{i-1} + \delta_i - (1+\nu)T_c = \delta_i - T_{i-1} - \nu T_c \tag{36}$$

and,

$$A_i = T_c + \Delta_i - (1+\nu)T_c = \Delta_i - T_{i-1} - \nu T_c \tag{37}$$

In addition, signals must arrive at the latest by the rising edge of the corresponding clock:

$$A_i \leq T_c - T_i \tag{38}$$

Long-Path Constraints: Combining (37) with the setup requirement (6), yields:

$$(1+\nu)T_c + T_{i-1} \geq \Delta_i + S_i \tag{39}$$

Combining (37) with (38 leads to another constraint:

$$(1+\nu)T_c + T_{i-1} - T_i \geq \Delta_i \tag{40}$$

Short-Path Constraints: Substituting (36) into the hold rquirement (5) yields:

$$\nu T_c + T_i \leq \delta_i - H_i \tag{41}$$

Solution: The feasible region for coincident n-phase clocking is defined in the $(n+1)$-dimensional space of clock variables by $5n$ linear inequalities:

- $2n$ long-path inequalities (39) and (40)
- n short-path inequalities (41)
- $2n$ minimum pulse-width inequalities (2) and (3)

These inequalities define a convex space, and the the minimum cycle time can be found by solving a linear program.

4 Examples and Results

We developed a computer program *pipe*T_c which determines the optimal cycle time for n-stage pipes. *pipe*T_c reads in the pipeline parameters (number of stages, stage delays, setup and hold times, and wave pipelining parameters) and produces the optimal clock schedules and signal waveforms for single-phase, conservative single-phase, and coincident multi-phase clocking using latches. We are using *pipe*T_c to study the relationship between the optimal cycle time and the various pipeline parameters. In this section we present two examples to highlight some of the issues in pipeline synchronization.

The first example is a 4-stage pipeline with a fairly uneven distribution of stage delays. Figures 8 and 9 show the single-phase feasible region, and the optimal clock schedules and signal waveforms at the inputs to all stages for two cases: (a) $H_1 = 2.0$, and (b) $H_1 = 2.5$. In both cases, $\nu = 0$. A summary of the results is also shown in Table 1. We make the following observations on these results:

1. The single-phase feasible region in Fig. 8 is non-convex. It consists of the shaded area in the T_c/T_1 plane as well as the line segment AB.
2. The optimal single-phase cycle time is the same as the optimal multi-phase cycle time (10.0), and is substantially lower than the conservative single-phase optimum (16.0) and the flip-flop optimum (18.0).
3. The single- and multi-phase solutions exhibit wave pipelining. In particular, during each clock cycle there are two wave fronts traveling in stage 0 (from $t = 2.0$ to $t = 8.0$) and in stage 2 (from $t = 2.0$ to $t = 4.0$). This observation is consistent with the earlier discussion in Sec. 2.3 since the delays of both stages 0 and 2 are greater than the cycle time. Examination of the signal waveforms suggests another way to determine if a given stage is wave pipelining: *stage i will "contain" 2 or more wave fronts of data in every clock cycle from $t = D_{i-1}$ to $t = A_i$; if $D_{i-1} \geq A_i$, then at most one wave front can be traveling in stage i.*

Case	H_1	1-Ph[1]		C 1-Ph[2]		4-Ph[3]	FF[4]
		T_c	T_1	T_c	T_1	T_c	T_c
(a)	2.0	10.000	8.000	16.000	2.000	10.000	18.000
(b)	2.5	16.500	1.500	16.500	1.500	10.125	18.000

[1] Optimal Single-Phase Solution

[2] Conservative Single-Phase Solution

[3] Coincident 4-Phase Solution

[4] Optimal Flip-Flop Solution

Table 1: Summary of Results for Example 1 (all times in units of nanoseconds)

4. When H_1 is increased from 2.0 to 2.5, the single-phase feasible region shrinks and becomes convex (Fig. 9). Now, the single-phase and the conservative single-phase solutions are identical ($T_{c,\min} = 16.5$), and both are larger than the multi-phase solution ($T_{c,\min} = 10.125$). Note that due to the non-convexity of the single-phase feasible region, a 0.5ns change in the hold time of stage 1 causes a 6.5ns change in the optimal cycle time. In contrast, the conservative single-phase and coincident multi-phase optima changed by 0.5ns and 0.125ns, respectively, in response to the same 0.5ns change in H_1. The flip-flop solution did not change.

The second example is a modification of the first in which the delay of stage 1 has been increased from 4.0 to 8.0. We study the effect of changing the hold time of stage 2 from 6.0 to 7.5 in 0.5ns increments. The results are shown in Figures 10, 11, 12, and 13, and are summarized in Table 2. The following additional observations can be made:

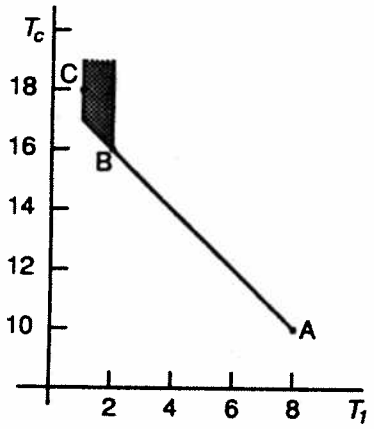
1. The single-phase feasible region is non-convex (Fig. 10), and becomes disconnected when H_2 is increased to 6.5 and 7.0 (Fig. 11 and Fig. 12). In particular, one of the disconnected subregions shrinks to a point. When H_2 is increased further to 7.5, the feasible region becomes convex (Fig. 13). Further increases in H_2 reduce the size of the feasible region, until it vanishes completely and the problem becomes infeasible.

2. The single-phase solution is not unique. In fact, for $H_1 = 6.0$ and $H_1 = 6.5$, the same optimal cycle time (11.0) can be achieved with a range of values for T_1. This situation will arise whenever the lower bound constraint on T_c given by (26) is active (this lower bound corresponds to the horizontal line segment).

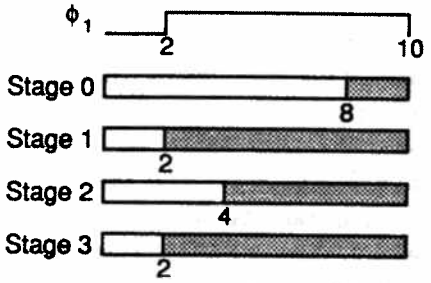
3. The conservative single-phase solution is now exhibiting wave pipelining. In fact, only the flip-flip solution is free from wave pipelining (recall that $\nu = 0$ in these experiments).

4. The 0.5ns increase in H_2 from 7.0 to 7.5 (cases c and d) causes a 2.5ns increase in the single-phase optimum, a 0.5ns increase in the conservative single-phase optimum, and a *one-sixth* ns increase in the multi-phase optimum; the flip-flop optimum does not change. This is consistent with the earlier observation in example 1. The curious one-sixth ns increase in the multi-phase case is readily explained in terms of the dual solution of the linear program used to find the optimal cycle time.

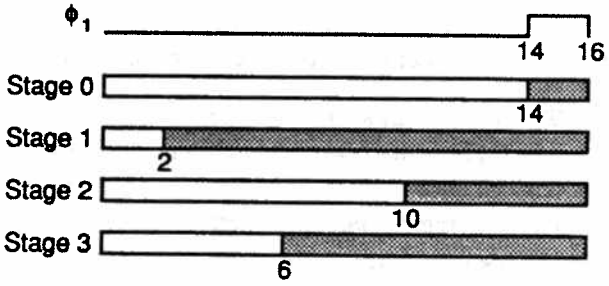
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	4.0	4.0	2.0	2.0
2	12.0	12.0	2.0	2.0
3	8.0	8.0	2.0	2.0



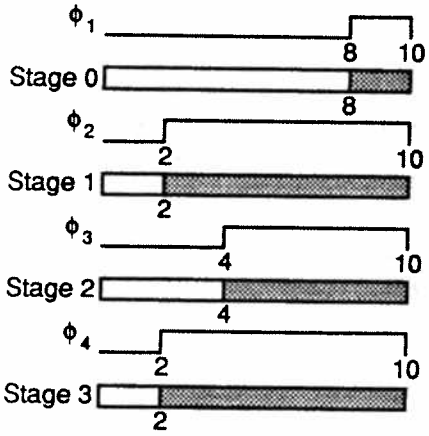
Single-Phase Feasible Region
(Latches)



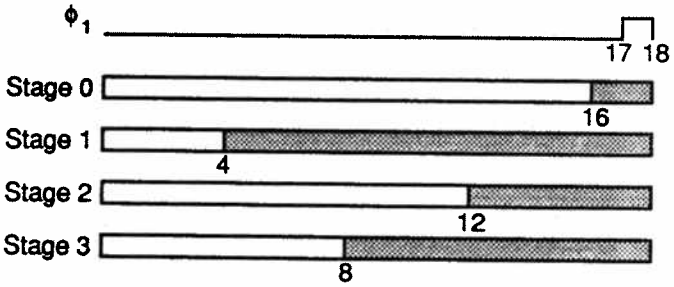
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



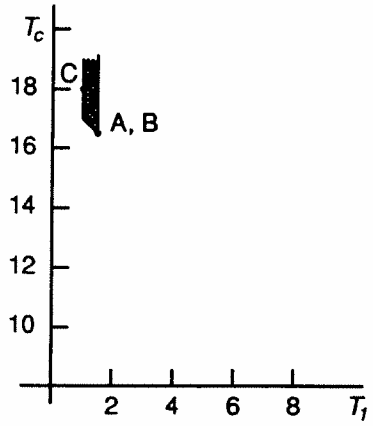
Optimal 4-phase Solution



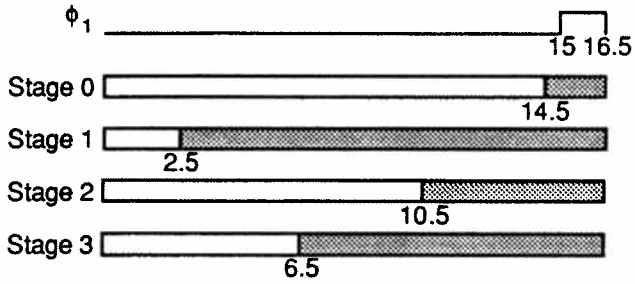
Optimal Flip-Flop Solution (Point C)

Figure 8: Example 1, Case (a) — $H_1 = 2.0$

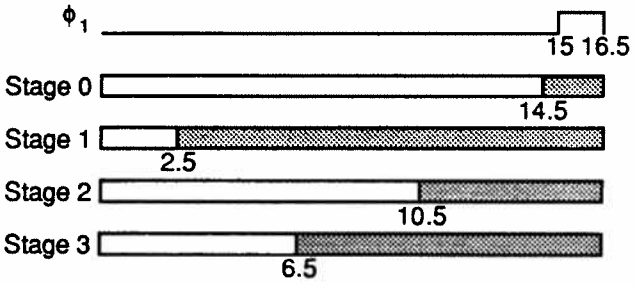
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	4.0	4.0	2.5	2.0
2	12.0	12.0	2.0	2.0
3	8.0	8.0	2.0	2.0



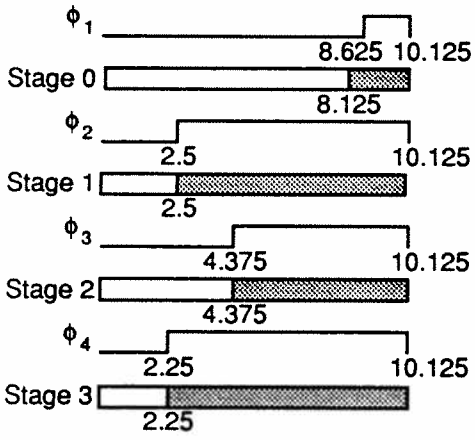
Single-Phase Feasible Region
(Latches)



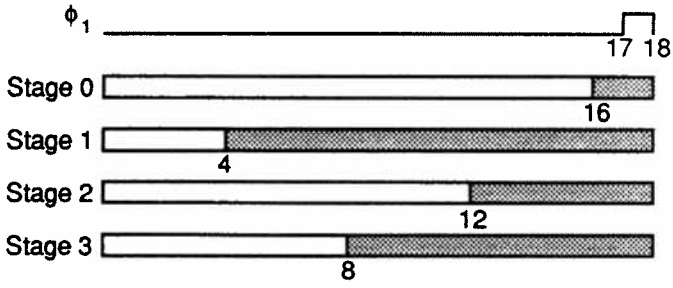
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



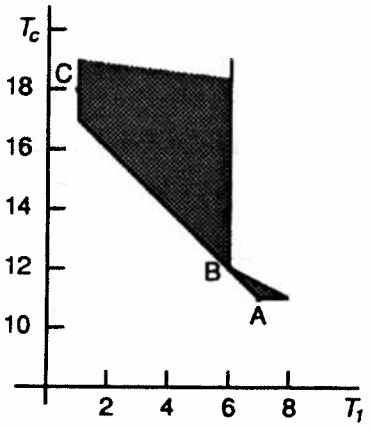
Optimal 4-phase Solution



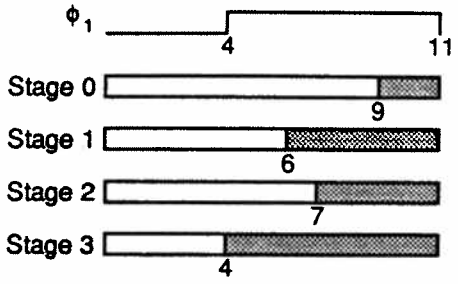
Optimal Flip-Flop Solution (Point C)

Figure 9: Example 1, Case (b) — $H_1^* = 2.5$

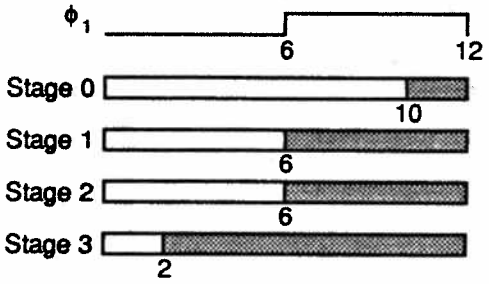
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	8.0	8.0	2.0	2.0
2	12.0	12.0	6.0	2.0
3	8.0	8.0	2.0	2.0



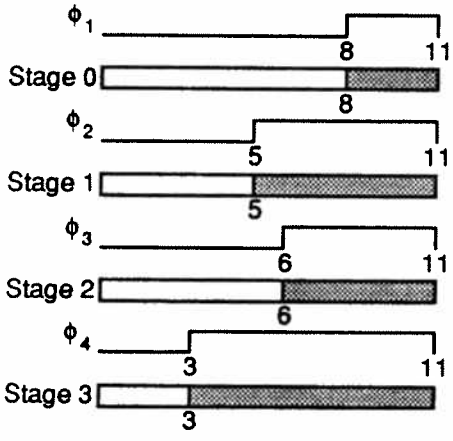
Single-Phase Feasible Region
(Latches)



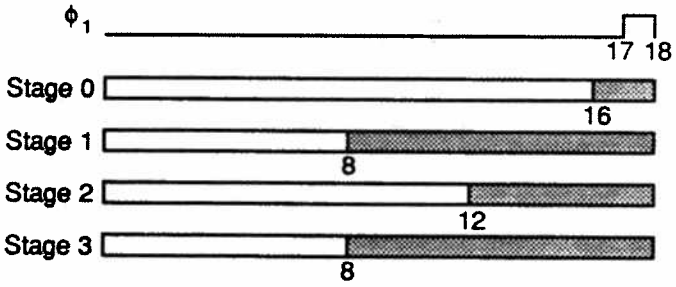
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



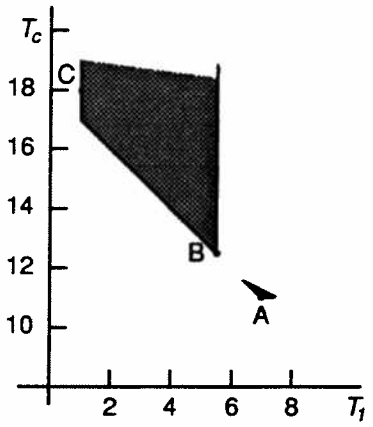
Optimal 4-phase Solution



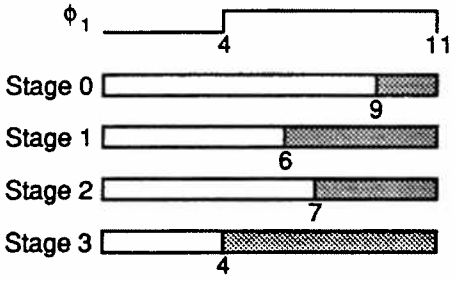
Optimal Flip-Flop Solution (Point C)

Figure 10: Example 2, Case (a) — $H_2 = 6.0$

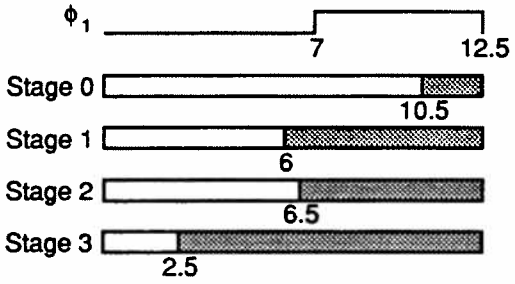
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	8.0	8.0	2.0	2.0
2	12.0	12.0	6.5	2.0
3	8.0	8.0	2.0	2.0



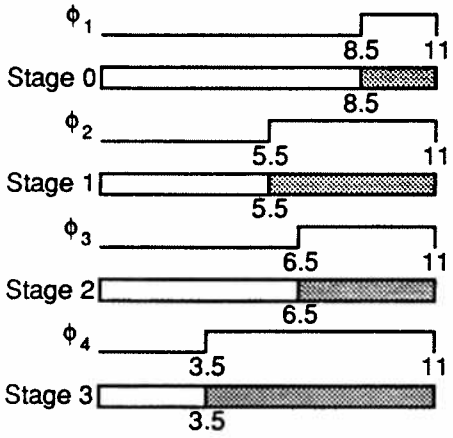
Single-Phase Feasible Region
(Latches)



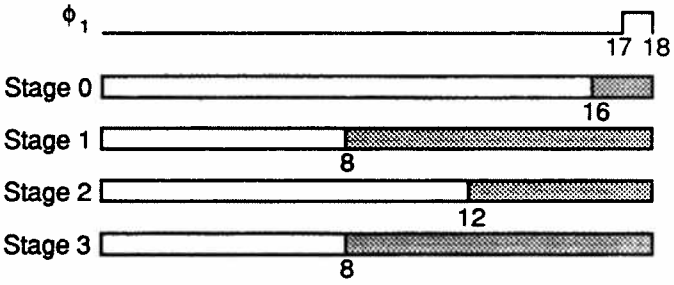
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



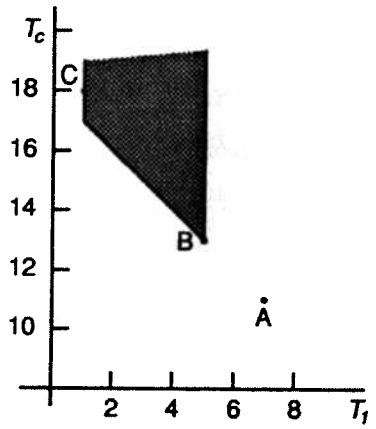
Optimal 4-phase Solution



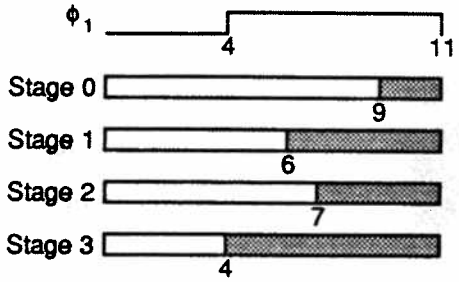
Optimal Flip-Flop Solution (Point C)

Figure 11: Example 2, Case (b) — $H_2 = 6.5$

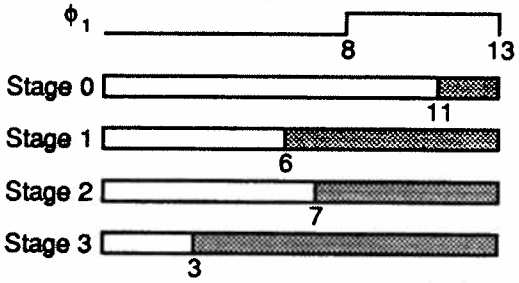
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	8.0	8.0	2.0	2.0
2	12.0	12.0	7.0	2.0
3	8.0	8.0	2.0	2.0



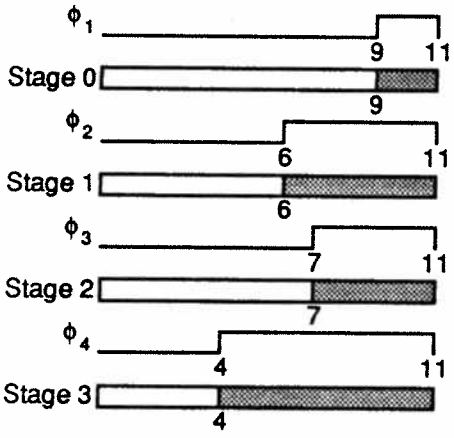
Single-Phase Feasible Region
(Latches)



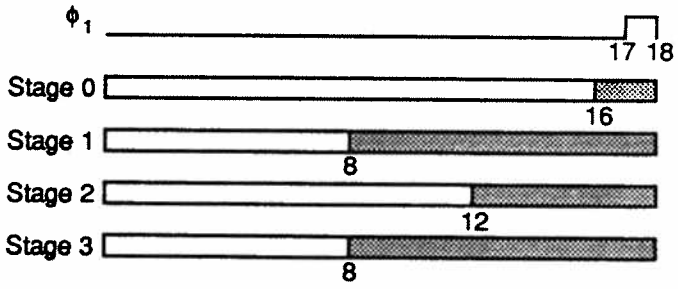
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



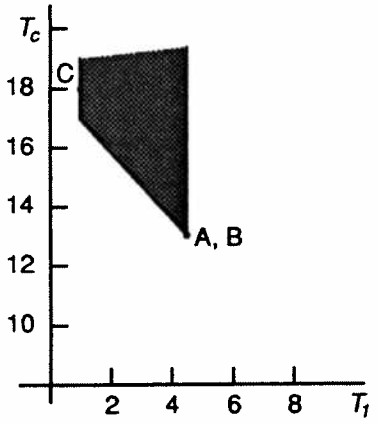
Optimal 4-phase Solution



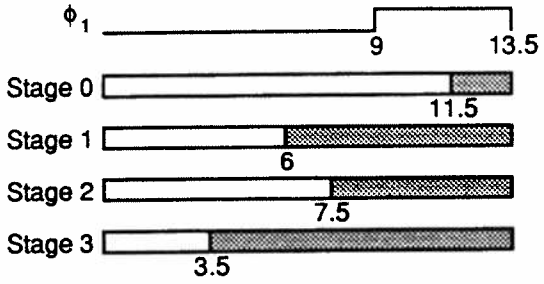
Optimal Flip-Flop Solution (Point C)

Figure 12: Example 2, Case (c) — $H_2 = 7.0$

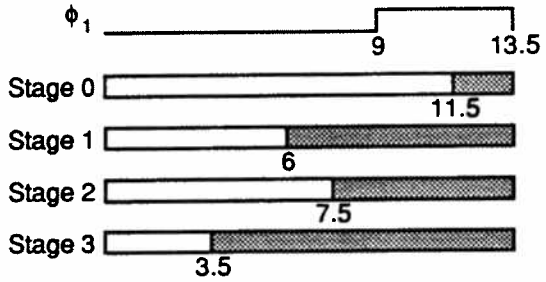
i	δ_i	Δ_i	H_i	S_i
0	16.0	16.0	2.0	2.0
1	8.0	8.0	2.0	2.0
2	12.0	12.0	7.5	2.0
3	8.0	8.0	2.0	2.0



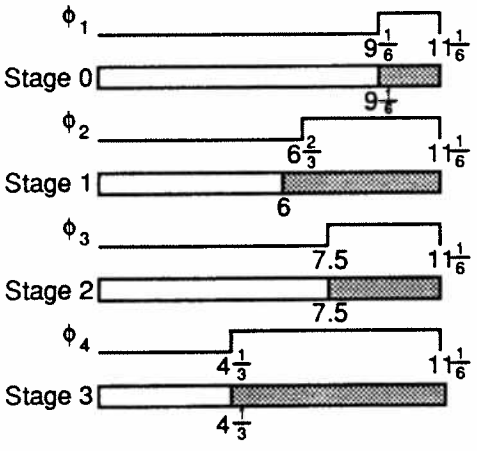
Single-Phase Feasible Region
(Latches)



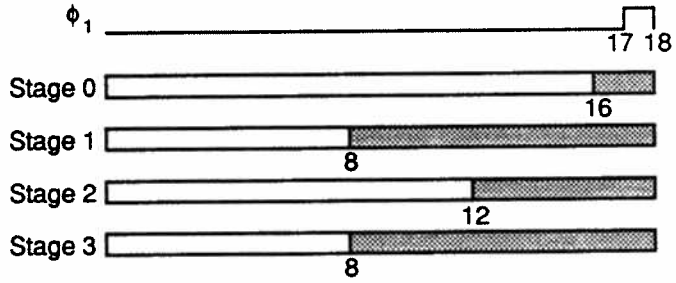
Optimal Single-Phase Solution (Point A)



Conservative Single-Phase Solution (Point B)



Optimal 4-phase Solution



Optimal Flip-Flop Solution (Point C)

Figure 13: Example 2, Case (d) — $H_2 = 7.5$

Case	H_2	1-Ph[1]		C 1-Ph[2]		4-Ph[3]	FF[4]
		T_c	T_1	T_c	T_1	T_c	T_c
(a)	6.0	11.000	[7.0, 8.0]	12.000	6.000	11.000	18.000
(b)	6.5	11.000	[7.0, 7.5]	12.500	5.500	11.000	18.000
(c)	7.0	11.000	7.000	13.000	5.000	11.000	18.000
(d)	7.5	13.500	4.500	13.500	4.500	11.167	18.000

[1] Optimal Single-Phase Solution

[2] Conservative Single-Phase Solution

[3] Coincident 4-Phase Solution

[4] Optimal Flip-Flop Solution

Table 2: Summary of Results for Example 2 (all times in units of nanoseconds)

5 Conclusions and Future Work

We have examined the problem of minimizing the cycle time for an n-stage pipeline under a variety of clocking conditions. One of the more interesting results of this examination is the fact that the feasible region for single-phase clocking maybe non-convex. We are currently investigating the practical ramifications of this result. We are also considering the effects of skew in the clock distribution network on the optimal cycle time, as well as its relationship to wave pipelining.

In general, minimizing the clock cycle time for a pipeline requires the consideration of clock generation and distribution, as well as logic and circuit design of the pipe stages. In this paper, we have assumed that the design of the pipe stage circuitry is fixed (i.e. the data propagation delays are specified) and that only the clock generator can be controlled to obtain a desired clock schedule. If the minimum cycle time corresponding to this "fixed" design is unacceptably high, a re-design (re-synthesis) would be necessary to reduce some or all of the delays. The results presented in this paper can be used to guide this re-synthesis step by identifying the most critical delays in the pipeline. We are currently investigating the integration of these results with a logic synthesis system.

References

[1] S. R. Kunkel and J. E. Smith, "Optimal Pipelining in Supercomputers," in *Proc. of the 13th Annual Symposium on Computer Architecture*, pp. 404–411, 1986.

[2] L. W. Cotten, "Circuit Implementation of High-Speed Pipeline Systems," in *AFIPS Fall Joint Computer Conference*, pp. 489–504, 1965.

[3] T. G. Hallin and M. J. Flynn, "Pipelining of Arithmetic Functions," *IEEE Trans. Computers*, vol. C-21, no. 8, pp. 880–886, August 1972.

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

[5] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "$checkT_c$ and $minT_c$: Timing Verification and Optimal Clocking of Synchronous Digital Circuits," in *ICCAD-90 Digest of Technical Papers*, Santa Clara, California, November 1990, IEEE.

[6] D. Wong, G. D. Micheli, and M. Flynn, "Inserting Active Delay Elements to Achieve Wave Pipelining," in *ICCAD-89 Digest of Technical Papers*, pp. 270–273, 1989.