GRAPH ALGORITHMS FOR EFFICIENT CLOCK SCHEDULE OPTIMIZATION

by

Narendra Shenoy, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/79

3 August 1992

GRAPH ALGORITHMS FOR EFFICIENT CLOCK SCHEDULE OPTIMIZATION

by

Narendra Shenoy, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/79

3 August 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

GRAPH ALGORITHMS FOR EFFICIENT CLOCK SCHEDULE OPTIMIZATION

by

Narendra Shenoy, Robert K. Brayton, and
Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M92/79

3 August 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Graph Algorithms for Efficient Clock Schedule Optimization

Narendra Shenoy, Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA-94720

1 Introduction

In aggressive designs, when the technology permits tight control on process variations, the performance of a design will probably rely on cycle borrowing using level-sensitive latches. At the same time, level-sensitive latches force the designer to obey short path constraints. Thus the use of level-sensitive elements is both a boon and a curse.

In this paper, we allow edge-triggered and level-sensitive memory elements in a circuit. A single clock, multi-phase clocking scheme is assumed. We assume the initial circuit is logically correct and are only concerned with optimizing the clock schedule, *i.e.* finding the minimum clock cycle time and the corresponding times for the rise and fall of all clock phases. Clocks are not gated and clock skew is assumed negligible. It is possible to extend the algorithms presented in this paper to incorporate non-zero skews.

Related early efforts in the area of timing issues ([2], [7], [4], [5]) concentrated on timing verification. Unger *et al.* ([9]) give an excellent description of the constraints for various clocking schemes used in digital design. Wallace *et al.* ([10]) use explicit unrolling of the clock for timing analysis. Weiner *et al.* ([11]) present an iterative scheme for timing analysis but do not take short paths into account. Dagenais *et al.* ([1]) use an iterative algorithm to calculate optimal clocking parameters. Ishii *et al.* ([3]) give an algorithm of polynomial complexity for verification of arbitrary clocking schemes but consider only the maximum propagation delays. Sakallah *et al.* ([6]) present an algorithm to optimize (conjectured to be optimum) the clock cycle based on linear programming. Szymanski ([8]) presents a restricted version of the model given in ([6]), makes a strong case for its use, and proposes an efficient technique to reduce the size of the linear programming problem.

The main contributions of this paper are efficient graph algorithms to compute an optimal clocking schedule of a circuit. Similar to [8] we make the restriction that all clock events be ordered. We also present in the appendix, a formal proof of the equivalence of an extension to the model used by Ishii *et al.* [3] and the restricted model of Sakallah [6]

given by Szymanski [8]. [1]

In section 2, we introduce the basic terminology. Section 3 deals with the clocking constraints. The results from Szymanski [8] are summarized in section 4. A simplified, albeit restricted, graph algorithm and its analysis is given in section 5. A general algorithm is presented in section 6. Section 7 presents the result of the algorithm applied to a video coder. Results concerning effectiveness and efficiency are given in section 8. Section 9 concludes the paper.

2 Definitions

Memory elements are assumed to be either edge-triggered or level-sensitive. Each memory element has a data input, a clock input and a single output. For an edge-triggered latch (flip-flop), at the appropriate edge of the phase connected to the clock input, the latch samples the data input and the value is presented at the output. This output remains stable until the next occurrence of the phase edge. Thus the input of a latch and its output are effectively decoupled. For a level-sensitive latch (also called just "a latch"), the data at the input is transmitted to the output as soon as the active period of the latch begins. The output is held at the data value from the time the active period ends until the next active period and fresh data from the input arrives. Thus the input and output are not isolated during the active period. The retardation([1]) at a level-sensitive latch is the amount by which the valid output is delayed since the beginning of the active period. It is the time borrowed from the current phase by the logic preceding the latch, in order to complete the computation. For correct operation of both memory elements, we need the data signal to be stable at the input before the latching edge occurs by an amount called the *set-up* time. It is also required that the signal be stable after the latching edge by an amount called the *hold* time.

We assume, without loss of generality, that edge-triggered elements sample input data on the falling transition and level-sensitive latches are active when the phase is high. Thus, the falling edge of each phase is the critical edge with respect to which set-up and hold constraints must be satisfied.

A clocking scheme, Φ is a collection of l periodic signals with a common period c, and is represented by $\Phi = (\phi_1, \phi_2, \cdots, \phi_l)$. Associated with each phase ϕ_i are two real numbers s_i and e_i, the time of occurrence respectively of the rising and falling edges of ϕ_i $(0 \leq (s_i, e_i) \leq c)$. Associated with each phase i is its local time zone, an interval of time of length c, such that the end of the active phase coincides with the end of the local time zone. Let $0 \leq e_1 \leq e_2 \cdots \leq e_l = c$; thus we choose the global reference time frame

[1] The extension to the Ishii's model consists of adding minimum propagation constraints. Though both forms have been freely used in the literature, we are unaware of any effort made to relate the two. The fact that the model proposed by Sakallah implies the constraints used by Ishii was shown in [8]. However, the converse was not known.

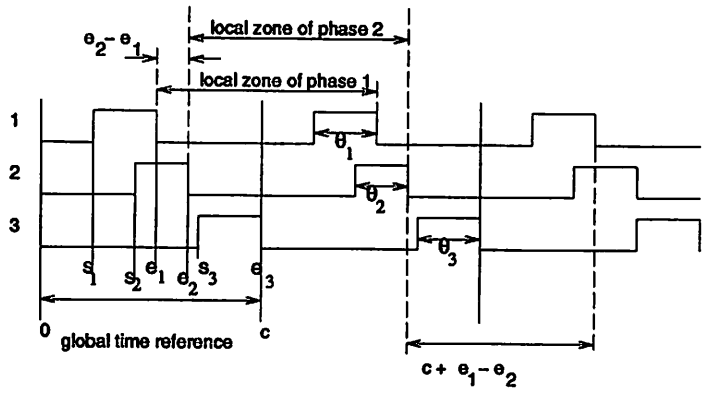

Figure 1: Three phase clocking scheme

as the last phase e_l. The clocking scheme specifies a complete ordering of the rise and fall of the phases. Note that this is a stronger assumption than in [6]. The reason we need to make this assumption will be clear in the next few sections.

We say $\phi_i \prec \phi_j$ if $e_i < e_j$. We use the phase shift operator E_{ij} introduced in [6], to translate all measurements of time from the local zone of phase i to the local zone of j. The phase shift operator for a path between phase i and phase j is defined as

$$\begin{aligned} E_{ij} &= e_j - e_i \qquad \text{if } \phi_i \prec \phi_j \\ &= c + e_j - e_i \qquad \text{otherwise} \end{aligned}$$

Consider the clocking scheme in Figure 1. Let an event be an upward or downward transition of a data signal. Consider an event at a ϕ_1 memory element occurring at time t_1, given in terms of the local time zone for ϕ_1. If this event causes another event at a ϕ_2 memory element with a delay say d, *i.e.* time $t_1 + d$ with respect to zone one; then E_{12} is the shift that must be subtracted from $t_1 + d$ to convert the event to the local time zone of ϕ_2. To distinguish between variables in the local time zone from the global frame we use a superscript L. Thus in the local time zone of ϕ_i, $e_i^L = c$. The local rise of a phase is $s_i^L = s_i + e_l - e_i = s_i + E_{il}$ if $s_i < e_i$ and $s_i^L = s_i - e_i = s_i + E_{il} - e_l$ if $s_i > e_i$. Henceforth we assume that $s_i < e_i$. If $s_i > e_i$ then the constraints (S2), derived in the next section will differ by a term as is explained in the Appendix. Consequently, *a priori* information is needed on the relative occurrence of the rise and fall of each phase in the global frame and the relative occurrence of the fall of all the phases. The first enables us to choose the correct form of the constraint from S2 and the second is necessary for the valid translation of events from one time zone to another.

The circuit C is modeled as a finite, edge-bi-weighted, directed multi-graph $G = (V, E, D, d)$. For every memory element $i \in C$ there is a vertex $v_i \in V$. In addition for every primary input and primary output of the circuit there is a vertex in V. If there

is a path of combinational logic from a memory element (or primary input), say i, to a memory element (or primary output), say j, we create an edge $e_{ij} \in E$ from v_i to v_j. The edge weight D_{ij} (d_{ij}) is the maximum (minimum) sum of the gate delays along the combinational paths from i to j. We say v_i is a fanin of v_j (v_j is a fanout of v_i) if there is a directed edge from v_i to v_j in the graph. We denote the fanin set of v_i by $FI(v_i)$, the fanout set by $FO(v_i)$. A path $v_i \xrightarrow{p} v_j$ is a set of alternating edges and vertices $\{v_i, e_{i1}, v_1, e_{12}, \cdots, e_{n-1j}, v_j\}$, and every pair of successive edges forms an input-output pair to the vertex between. Henceforth all vertices will be simply indexed by a variable instead of a variable subscript of v, *i.e.* we will use i to denote a vertex instead of v_i. We denote the phase of the latch represented by vertex v_i as $\phi(i)$.

The rise (fall) of the phase to memory element i is denoted by s_i (e_i) when there is no ambiguity. So latches i and j may have the same phase k and e_i and e_j will both denote e_k. Thus the symbol e_i is overloaded to signify both the fall of phase i and the fall of the phase used to clock memory element i.

Each latch in the circuit is associated with four variables:

$$\begin{aligned} A_i &= \text{latest that the signal is valid at the input of latch } i, \\ a_i &= \text{earliest that the signal is valid at the input of latch } i, \\ R_i &= \text{latest that the signal is valid at the output of latch } i, \\ r_i &= \text{earliest that the signal is valid at the output of latch } i. \end{aligned}$$

These variables are measured with respect to the local time zone of the phase of the latch. We constrain all variables to lie within $[0, c]$. The problem posed is:

Given a circuit G(V, E, D, d), find the minimum clock cycle c, and the rise and fall times for each of the phases (s_i, e_i), so that the clocking scheme meets all timing constraints.

3 Clocking Constraints

The clocking constraints for optimal clock schedules may be written as follows [6] [8]. We use the conservative design constraints. Szymanski [8] makes a strong case for the use of this set of constraints versus the aggressive set described by Sakallah *et al.*. The long path constraints are of the form:

$$A_i = \max_{\forall j \in FI(i)} (R_j + D_{ji} - E_{ji}). \tag{1}$$

The short path constraints are of the form:

$$a_i = \min_{\forall j \in FI(i)} (r_j + d_{ji} - E_{ji}). \tag{2}$$

The propagation constraints for a latch are:

$$R_i = \max(A_i, s_i^L) \tag{3}$$

$$r_i = s_i^L. \tag{4}$$

(Sakallah used $r_i = \max(a_i, s_i^L)$ instead) [2]. The constraints for a flip-flop become:

$$R_i = e_i^L \tag{5}$$

$$r_i = e_i^L. \tag{6}$$

We also need

$$a_i \leq A_i \tag{7}$$

The set-up and hold constraints are of the form

$$A_i \leq e_i^L - S \tag{8}$$

and

$$H \leq a_i \tag{9}$$

where S is the set-up time and H is the hold time for a latch/flip-flop. Let this set of constraints (1-9) be called S1.

We now derive a set of inequalities which are both necessary and sufficient for correct clocking. Denote a flip-flop by F and a latch by L. We can have 4 types of paths in the graph G. The path $j \xrightarrow{p} i$ can be

1. from a latch j to a latch i (LL)
2. from a latch j to a flip-flop i (LF)
3. from a flip-flop j to a latch i (FL) or
4. from a flip-flop j to a flip-flop i (FF).

In addition we constrain all paths to necessarily terminate if they encounter a flip-flop. Let the set of all such paths be denoted by $\mathcal{P}(G)$. For a path p we index the vertices on the path with $p_0(= j), \cdots, p_N(= i)$. A path is allowed to have repeated vertices; hence the set $\mathcal{P}(G)$ can be infinite.

Theorem 1: The set of constraints S1 described above are equivalent to the following inequalities (S2) for each path $p \in \mathcal{P}(G)$. The path p is denoted by $j \xrightarrow{p} i$. The inequalities depend on the type of p:

[2] It is the relaxation of this constraint that prevents the min and max interactions between the constraint in (2) and (4)

1. p is of type LL or LF: $e_i \geq s_j + D_{ji} - K^p_{ji}c + S$

2. p is of type FL or FF: $e_i \geq e_j + D_{ji} - K^p_{ji}c + S$ where $D_{ji} = \sum_{k=1}^{N} D_{k-1k}$ and K^p_{ji} is an integer defined recursively (from 1 to N) as

$$K^p_{jk} = \begin{cases} 0 \text{ if } k = j \text{ (i.e. } k = p_0) \\ K^p_{jk-1} \text{ if } e_{k-1} < e_k \\ K^p_{jk-1} + 1 \text{ if } e_k \leq e_{k-1} \end{cases} \tag{10}$$

In addition for each path $p : j \xrightarrow{p} i$ consisting of a single edge(e_{ji}) we need

3. p is of type LL or LF: $e_i \leq s_j + d_{ji} + (1 - K^p_{ji})c - H$

4. p is of type FL or FF: $e_i \leq e_j + d_{ji} + (1 - K^p_{ji})c - H$

where $d_{ji} = \sum_{k=1}^{N} d_{k-1k}$. Constraints 1-2 are called the set-up constraints of S2 and 3-4 called the hold-time constraints. If $e_j < s_j$ then we need to subtract c from the right hand side of the equations.

The proof of Theorem 1 is given in the Appendix.

4 Redundant Constraints

This section is basically a reproduction of the equivalent theorems from [8], that can be used to reduce the number of constraints in S2. Constraints 3 and 4 in S2 total up to $|E|$ constraints. Constraints 1 and 2, on the other hand, must hold for all paths, and possibly are infinite in number. Hence we must find a more compact representation. Consider the constraints 1 and 2 in S2. These can be represented in the form $x_i - x_j \geq \alpha^p_{ji} - K^p_{ji}c$, where the parameters α^p_{ji}/K^p_{ji} depend on the path $p : u \rightarrow v$ ($\phi(u) = j$, $\phi(v) = i$) and $x_i = e_i$ or s_i, $x_j = e_j$ or s_j.

$$\alpha^p_{ji} = \sum_{k=1}^{N} D_{k-1k} + S \qquad k = 0 \text{ is } i \text{ and } k = N \text{ is } j \tag{11}$$

and $K^p_{ji} \geq 0$. The dependence of the constraints (on the path) is over all paths from memory element j to memory element i. We modify this dependence to include all paths of the same type (LL/LF or FL/FF) as p and having the same phases at the end points. Let $U^c_{ji} = \max_{p \in \mathcal{P}(\mathcal{G})} (\alpha^p_{ji} - K^p_{ji}c)$. For a given clock cycle c, U^c_{ji} is the relevant bound. Let C be the set of simple cycles in $\mathcal{P}(G)$. Recall that there must be at least one memory element in every cycle. Consequently for any cycle $p : i \rightarrow i$, $K^p_{ii} = K_p > 0$. The vertex given

by $k = 0$ is the same as the vertex given by $k = N$. A weak lower bound on the clock cycle c is

$$c \geq \psi \equiv \max_{p \in \mathcal{P}(\mathcal{G})} \left(\frac{\sum_{k=1}^{N} D_{k-1k}}{K_p} \right). \tag{12}$$

Consequently $U_p^c - S \leq 0$ for all paths that contain a simple cycle and $c \geq \psi$. As a result we conclude that a simple way to compute U_{ji}^c is

$$U_{ji}^c = \max_{\forall \text{ simple paths } p:\, u \rightarrow v,\, \phi(u) = j,\, \phi(v) = i} (\alpha_{uv}^p - K_{uv}^p c). \tag{13}$$

Intuitively, if we encounter any cycle on a path from u to v, we can only decrease U_{ji}^c. Also note that as c increases, U_{ji}^c decreases for all paths from phase j to phase i. However, if we make use of the lower bound on c, we will get a set of tight constraints. Hence $U_{ji}^c \leq U_{ji}^{\psi}$ for any feasible clock cycle c.

For a given clock cycle c, the relevant constraints S_c may be obtained as follows. Each edge e in the graph G is weighted with $D_e - K_e c$. Solve an all pairs longest path problem (with the constraint that a path cannot continue if it encounters a flip-flop), given that the graph has no positive cycles. The length of the longest path gives us the constant on the right hand side of the inequality, denoted by U_{ji}^c. Since we have l phases, we can have at most $2l$ variables (rise and fall times). The value of K_{ji}^p can range from 0 to $|V|-1$. Hence we have $|V|l(2l-1) = O(|V|l^2)$ relevant constraints for the circuit. If we are given a clock cycle c this reduces to $O(l^2)$ constraints. To compute all relevant long path constraints for all valid clock cycles, we use the approach suggested by Szymanski [8]. This has a complexity of $O(l|E||V|)$. This approach also requires the computation of ψ ($O(|V||E|b)$, where b is the number of bits of accuracy required in computing ψ).

The constraint 5 to 8 in S2 can be written as $x_i - x_j \leq \gamma_{ji}^e + \delta_{ji}^e c$, with

$$\gamma_{ji}^e = d_{ji} - H \qquad e_{ji} \in E \tag{14}$$

and

$$\delta_{ji}^e = 1 - K_{ji}^e. \tag{15}$$

Once again we modify this dependence to include all paths of the same type (LL/LF or FL/FF) as p and having the same phases at the end points. Recall that for an edge e_{ji}, $K_{ji} \in \{0, 1\}$, implying $\delta_{ji}^e \in \{0, 1\}$. Let $L_{ji}^c = \gamma_{ji}^e + \delta_{ji}^e c$. Thus L_{ji}^c decreases with decreasing c, since $\delta_{ji}^e \geq 0$. With the lower bound on c, we get $L_{ji}^c \geq L_{ji}^{\psi}$ for any feasible clock cycle. Since we have E constraints, we can just keep the minimum value of γ_{ji} for $K_{ji} \in \{0, 1\}$. This yields $2l^2$ constraints.

So in all we have $O(|V|l^2)$ constraints.

5 Optimization

Sakallah *et al.* [6] formulated the clock cycle minimization as a Linear Programming (LP) problem using relaxed constraints derived from S1. Szymanski used techniques to reduce the number of constraints but still solved it as a LP. In this section we show that the LP has a special structure that makes it possible to solve it efficiently.

The problem formulation for clock cycle optimization may be posed in several equivalent forms. The Linear Programming formulation is conceptually the simplest.

$$
\begin{array}{lll}
P: & \min(c) & \\
s.t. & x_i - x_j \leq -\alpha_{ji} + K^p_{ji}c & O(|V|l^2)\text{constraints} \\
& x_i - x_j \leq \gamma_{ji} + \delta_{ji}c & O(2l^2)\text{constraints} \\
& 0 \leq x_i \leq c & \\
& \psi \leq x_0 (= e_l) = c &
\end{array}
$$

We could solve this as a Linear Programming problem alone. Instead we propose a binary search method. If we fix the clock cycle to a value say $c = \pi$, then the constraint set reduces to $O(l^2)$. This can be done by just evaluating the right hand sides with $c = \pi$ and picking the minimum. This takes $O(|V|l^2)$. The constraints are then of the form

$$
\begin{array}{ll}
x_i - x_j \leq k^\pi_{ji} & O(l^2)\text{constraints} \\
x_i - x_0 \leq 0 & O(l)\text{constraints} \\
0 \leq x_i & \\
\psi \leq x_0 (= e_l) = \pi &
\end{array}
$$

We can choose to append the phase separation constraints at this time. A restricted set of duty cycle constraints are also permitted. The next section will present a generalized algorithm which will handle arbitrary duty cycle constraints. This reduces to a feasibility check at $c = \pi$, and the theorem below shows that this can be done in polynomial time.

Theorem 2: Given a clock cycle $\pi > \psi$, it is possible to check if π is a valid clocking scheme, and if so to find values for the rise and fall times of the l phases in $O(l^3)$ time.

Proof: It is well known that the feasibility of a set of constraints of the form $x_i - x_j \leq k_{ji}, k_{ji} \in R$, can be related to the shortest path on a graph problem.

To check for feasibility we construct a graph $G_p(V_p, E_p)$ as follows. For each variable x_i construct a vertex v_{p_i}. For each constraint $x_i - x_j \leq k^\pi_{ji}$, construct an edge from v_{p_j} to v_{p_i} with weight k^π_{ji}. Henceforth when we say "we add an edge of weight w", we mean the following – if a previous edge exists we simply change its weight to be the minimum of the original weight and w. If no such edge exists we create a new edge of weight w in the graph. Add edges to all vertices $v_{p_i}, (i \neq 0)$ from v_{p_0} with weight 0 ($v_{p_i} \leq v_{p_0}$). Construct a zero vertex v_z (not to be confused with v_{p_0}) which has edges from all vertices

other than v_{p_0} with weight zero ($v_{p_i} \geq 0$). Weigh the edge from v_{p_0} to v_z with $-\pi$. Add an edge from v_z to v_{p_0} of weight π. This construction makes the graph G_p strongly connected. From every vertex there is an edge to v_z. There is an edge from v_z to v_{p_0}, and there is an edge from v_{p_0} to all other vertices.

Initialize the potential of v_z to 0 and the potentials of all other nodes to $+\infty$. Now do a Bellman-Ford iteration for the shortest paths. If there is a negative cycle in G_p, it will be detected and such a cycle implies a set of inconsistent equations, implying infeasibility. Else, the algorithm will terminate with a set of consistent potentials for all vertices. The complexity is $O(l^3)$. If there are any upper bounds on variables, we initialize the potential of the vertex that represents that variable to the upper bound instead of $+\infty$.

If there is a negative cycle in the graph G_p, it implies that the constraints are infeasible. Let C_- be a negative cycle with weight $-W$ through, vertex v_{p_i}. This implies we have a constraint (after elimination from the set) $x_i \leq x_i - W$, *i.e.* $0 \leq -W$, clearly infeasible. □

In order to do guarantee that binary search will find the optimum clock cycle, we need to prove two results:

1. Convexity of P: If x^q, x^r are feasible to the problem P with clocks cycles q and $r(q > r)$ respectively, then there exists a solution to all clock cycles between r and q.

2. There is a tight upper bound on the clock cycle. Note that this implies that the upper bound is actually attained. So we have to show the existence of an upper bound C and a feasible solution x^C for $c = C$.

Theorem 3: Let x^q, x^r be feasible solutions to the constraints in P with clocks cycles q and r respectively $(q > r)$. Let $c = \lambda q + (1-\lambda)r (\lambda \in [0,1])$. Then $x = \lambda x^q + (1-\lambda)x^r$ is a feasible solution to the constraints in P with clock cycle c.
Proof: Consider any constraint of the form $x_i - x_j \leq a + bc$, where a, b are real constants. We know

$$\begin{aligned}
x_i^q - x_j^q &\leq a + bq \\
x_i^r - x_j^r &\leq a + br \\
\lambda x_i^q - \lambda x_j^q &\leq \lambda a + \lambda bq \\
(1-\lambda)x_i^r - (1-\lambda)x_j^r &\leq (1-\lambda)a + (1-\lambda)br \\
(\lambda x_i^q + (1-\lambda)x_i^r) - (\lambda x_j^q + (1-\lambda)x_j^r) &\leq a + b(\lambda q + (1-\lambda)r)
\end{aligned}$$

Hence $x = \lambda x^q + (1-\lambda)x^r$ is feasible for clock cycle $(\lambda q + (1-\lambda)r)$.
Theorem 4: $C = |V| 2l \max_{e_{ij} \in E} D_{ij}$, where D_{ij} is the delay between vertex i and vertex j is a tight upper bound on the clock cycle.
Proof: We will give an algorithm to find C. The short path constraints (S2:5-8) have a positive right hand side always. The long path constraints have a right hand side of the

form $-\alpha_{ji} + K_{ji}^p c$ with K_{ji}^p, possibly equal to zero. However any cycle in G_p must have at least one edge going from a phase ϕ_i to a phase ϕ_j such that $e_j \leq e_i$. Such an edge will have a strictly positive coefficient for c, *i.e.* K_{ji}^p or δ_{ji} will be > 0. The smallest value for this coefficient is 1. The largest value α_{ji} can have for an edge is $|V| \max_{e_{ij} \in E} D_{ij}$. Any cycle can have at most $2l$ edges. So for a clock cycle $C \geq 2l|V| \max_{e_{ij} \in E} D_{ij}$ there will be no negative cycles in the graph G_p. Now carry out the Bellman-Ford iteration for $c = C$. Since there are no negative cycles, we are guaranteed that the algorithm will converge to a valid solution. □

Note that we never really need to compute C, only justify the existence of C that depends on the delays of the gates in the circuit.

Lemma 1: The complexity of binary search is $O((|V|l^2+l^3)\log C) \sim O(|V|l^2 \log |V|)$[3].

Proof: The first term is the complexity of selecting the minimum value of the right hand side l_{ji}^{π} for a given clock cycle π. The second term is due to the Bellman-Ford iteration in Theorem 3. If we normalize all numbers by $\max_{e_{ij} \in E} D_{ij}$, then for $|V| \gg l$ we get $\log C = \log(2l|V|) \sim O(\log |V|)$. Hence $O((|V|l^2 + l^3)\log C) \sim O(|V|l^2 \log |V|)$. □

It should be pointed out that duty cycle constraints are of the form $mc \leq x_i - x_j \leq Mc$, where $0 \leq m \leq M \leq 1$. The minimum duty cycle constraint will cause the coefficient for c to be negative, *i.e.* $-m$. Theorem 4 excludes the presence of such constraints. Consequently the bound in the previous Lemma does not hold in the presence of minimum phase separation constraints. In fact the addition of these constraints may render the problem infeasible. Intuitively, a long path may force the clock cycle c to be of at least value Π; this forces the on-time of a phase to be at least $m\Pi$, and a short path may cause a violation. The next section describes a general algorithm which finds the optimum clock cycle or detects infeasibility. We are guaranteed that the binary search will work only if there is no cycle for which the sum of the coefficients of c is negative. A simple case is when $m \in [0, \frac{1}{7}]$.

6 Optimization: A General Algorithm

This algorithm is motivated by the technique used in Linear Programming which adds a constraint to the active set only when needed. We take advantage of the special structure of the constraints (Theorem 2) to find a feasible solution for a given clock cycle, if it exists. The general problem is of the form

$$\begin{aligned} P: \quad & \min(c) \\ s.t. \quad & x_i - x_j \leq \min_{k=1,\ldots,N}(a_{ji}^k + b_{ji}^k c) \\ & c_L \leq c \end{aligned}$$

[3] log is logarithm to base 2.

We construct a constraint graph $G_p(V_p, E_p)$ as described in Theorem 2. The general algorithm is described below.

General Algorithm

```
c = clock cycle
c_L = lower bound on c
G_p(V_p, E_p) = constraint graph
c = c_L
while (TRUE) {
      flag = check-constraints(G_p, c)
      if (flag == ALL POSITIVE CYCLES) {
            return TRUE (c is the optimum clock cycle)
      }     if (flag == NEGATIVE CYCLE) {
            c = new-lower-bound
      }     if (flag == INFEASIBLE) {
            return FALSE (problem is infeasible)
      }
}
```

The routine `check-constraints()` for a given clock cycle can return one of three values:

1. `ALL POSITIVE CYCLES`: The set of constraints for the current clock cycle is feasible.
2. `NEGATIVE CYCLE`: The set of constraints for the current clock cycle is infeasible because at least one negative cycle exists in G_p.
3. `INFEASIBLE`: The problem is infeasible.

The search starts at the lower bound of the clock cycle. The routine `check-constraints()` evaluates the dominating constraint for each edge e_{ji} *i.e.* $\min_{k=1,\cdots N}(a_{ji}^k + b_{ji}^k c)$ and setting it as the edge weight w_{ji}. Floyd-Warshall is used detect the shortest path from x_p to x_q, keeping track of the sum of the b_{ij}'s for the shortest path. During the Floyd-Warshall iterations we keep track of the diagonal values of the Floyd-Warshall matrix. As soon as one of these values becomes negative, we analyze all the cycles detected so far for each vertex. Let $W_{ii}^c = \sum(w_{jk})$ denote the weight of the cycle if one exists from x_i to x_i at this time. Let B_{ii} denote the sum of the b_{ij}'s for the cycle. There arise four cases as shown in figure 2:

1. $W_{ii}^c < 0$ and $B_{ii} > 0$: Feasible clock cycles must be greater than or equal to $c + \frac{-W_{ii}^c}{B_{ii}}$ = (new-lower-bound)(cycle C_1 in figure).
2. $W_{ii}^c < 0$ and $B_{ii} \leq 0$: The problem is infeasible because, for every clock cycle greater than c this cycle will have a negative value (cycle C_2 in figure).

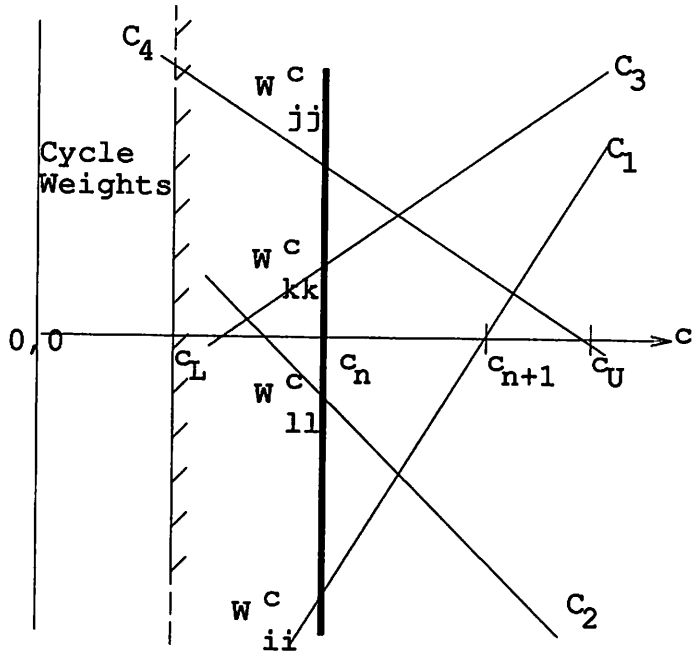

Figure 2: Graphical interpretation of the cycle weights

3. $W_{ii}^c \geq 0$ and $B_{ii} \geq 0$: c is possibly feasible if $W_{jj}^c \geq 0$ holds for all vertices j and the Floyd-Warshall algorithm is completed (cycle C_3 in figure).

4. $W_{ii}^c \geq 0$ and $B_{ii} < 0$: Feasible clock cycles must be less than or equal to $c + \frac{W_{ii}^c}{-B_{ii}}$ (cycle C_4 in figure).

Note that the clock cycle being tested (c) is monotonically increasing. If we encounter a vertex i satisfying case 1, then we get a lower bound on the clock cycle. If vertex i satisfies case 2 then the problem is infeasible. Cases 3 and 4 does not give us any information unless the regarding infeasibility because we are examining just one of the cycles in the graph. The last case does give an upper bound on the clock cycle, which can be used to detect infeasibility early. The ensuing lemmas provide insight into the problem.

Lemma 2: If for any $c(\geq c_L)$, there is a negative cycle through vertex i, such that $W_{ii}^c < 0$, and $B_{ii} \leq 0$ (for that cycle), then for all $c' \geq c$, there is a negative cycle through i.

Proof: Let us denote the negative cycle through i at clock cycle c as C_{-}. Let $\hat{k}$ be the

dominating constraint for each edge e_{ij} in C_- at clock cycle c [4]. Then

$$\begin{aligned}
\min_{k=1,\cdots,N}(a_{ij}^k + b_{ij}^k c') &\leq a_{ij}^{\hat{k}} + b_{ij}^{\hat{k}} c' \\
\sum_{C_-}(\min_{k=1,\cdots,N}(a_{ij}^k + b_{ij}^k c')) &\leq \sum_{C_-}(a_{ij}^{\hat{k}} + b_{ij}^{\hat{k}} c') \\
&\leq \sum_{C_-}(a_{ij}^{\hat{k}} + b_{ij}^{\hat{k}} c) + b_{ij}^{\hat{k}}(c' - c) \\
&< W_{ii}^c + B_{ii}(c' - c) \\
&< 0
\end{aligned}$$

Thus the weight of C_- for a clock cycle c' is negative. □

Lemma 3: If the problem P is infeasible then $\exists$ a cycle C in G_p, such that

1. $\sum_{e_{ij} \in C, k=1,\cdots,N} (b_{ij}^k)$ is strictly negative, or
2. $\sum_{e_{ij} \in C, k=1,\cdots,N} (a_{ij}^k) < 0$ and $\sum_{e_{ij} \in C, k=1,\cdots,N} (b_{ij}^k) = 0$.

Proof: By contradiction. Suppose the above condition does not hold *i.e.*, for all cycles if $\sum_{e_{ij} \in C, k=1,\cdots,N} (a_{ij}^k) < 0$, then $\sum_{e_{ij} \in C, k=1,\cdots,N} (b_{ij}^k) > 0$, else $\sum_{e_{ij} \in C, k=1,\cdots,N} (b_{ij}^k) \geq 0$. Then for a sufficiently large c, it is possible to make all cycles in the graph have strictly positive weight and a feasible solution to P can be found using Theorem 2. □

Theorem 5: The General Algorithm is complete, *i.e.* it finds an optimum solution if one exists, else it reports the problem as infeasible.

Proof: The proof is given graphically. We break the proof into two parts; in the first part we prove that the algorithm converges to a solution in a finite number of iterations, second we prove that the clock cycle is optimum.

Let c_n denote the value of the c in the n^{th} iteration, $n \geq 0$, $c_0 = c_L$. The proof relies on the fact that the number of cycles in the constraint graph is finite, though exponential in the number of constraints (number of cycles $\sim O((N+1)^{|E_p|})$. Assume without loss of generality, that there negative cycles $(C_0, C_1, \cdots C_n)$ in the constraint graph for $c_0, c_1, \cdots, c_n$. Note that cycle C_p is negative for all values of $c \in [c_p, c_{p+1}]$. When the algorithm reports the existence of a negative cycle in iteration n, it either gives a value for c_{n+1} in the next iteration, or reports the problem to be infeasible. The proof for infeasibility is provided by a set of cycles such that the minimum weight of these cycles for all $c \geq c_L$ is strictly negative.

[4] *i.e.* $a_{ij}^{\hat{k}} + b_{ij}^{\hat{k}} c = \min_{k=1,\cdots,N}(a_{ij}^k + b_{ij}^k c)$

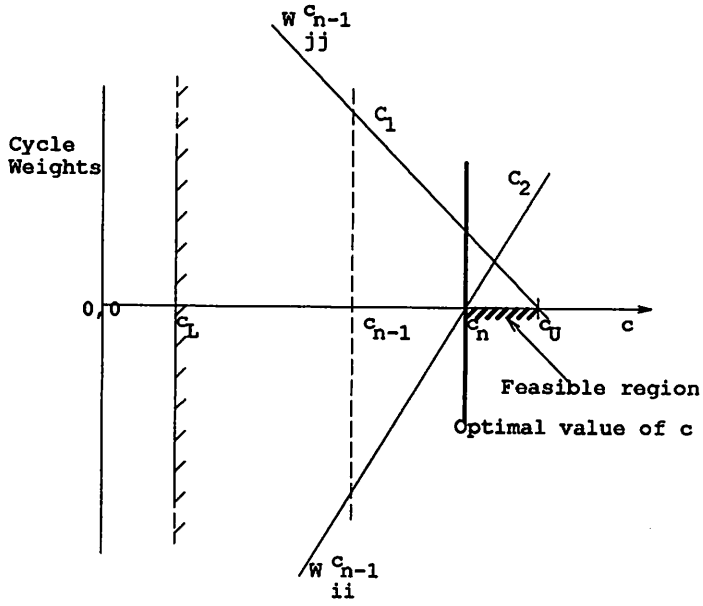

Figure 3: Graphical interpretation of optimality

To show that the clock cycle reported in iteration n (if the problem is feasible) is optimum, we note that for all $c \in [c_L, c_n)$, we have a proof of infeasibility, namely a set of negative cycles. For $c = c_n$, there are no negative cycles, hence there is a solution to the x_i's from Theorem 2. Intuitively the feasible region is the interval defined by the cycles determining the new upper and lower bounds on c (C_1 and C_2 in figure 3. □

7 Example

In this section we describe the algorithm for optimal clocking applied to the data path of a signal processing design taken from [12]. The data path is a part of a video data compression system. It uses a delta PCM compression algorithm. The compression is achieved by a non-linear quantization operation Q. Its inverse D is used to maintain the prediction value. The circuit is shown in figure 4. The delays of each component are shown along side ([min, max] delays). The input signal is 9 bits and the compressed value is 6 bits. All 9 bit lines are shown as dark lines and the 6 bit lines are light. The memory elements at the inputs to the subtraction unit are flip-flops (falling edge). The rest of the elements are level-sensitive (active high). The circuit graph for this circuit is shown in figure 5.

For this circuit, a lower bound on the clock (using equation 14) is 60 units (cycle through `b, c, f, b`). However the optimum clocking for this circuit is 88 units due to the

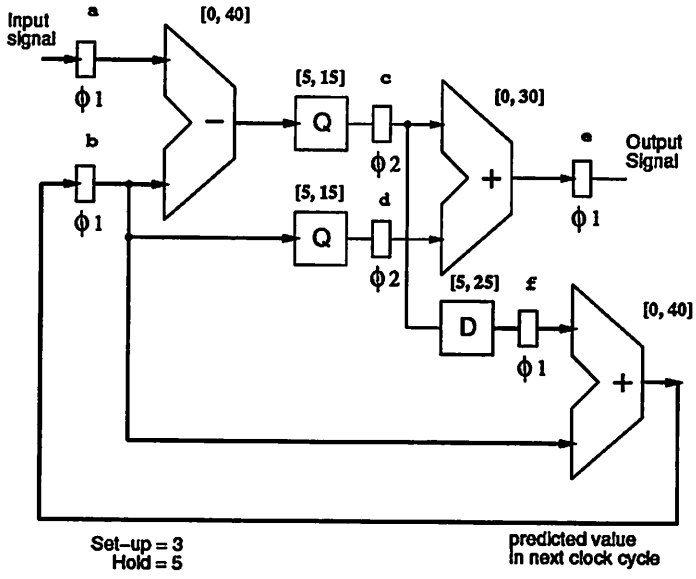

Figure 4: Video Coder

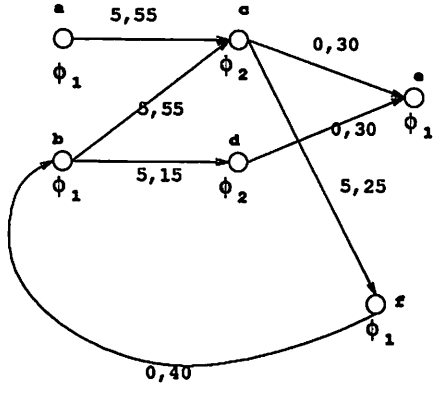

Figure 5: Circuit graph for Video Coder

path b, c, e. A valid optimum solution is $s_1 = 27$, $e_1 = 30$, $s_2 = 85$ $e_2 = 88 = c$.

8 Experiments

We use several MCNC sequential examples and some of the largest ISCAS circuits as benchmarks. These were originally intended to be single phase edge-triggered designs. In order to construct two-phase sequential circuits, we use a technique due to Szymanski [8]. First replace all edge-triggered elements by level-sensitive ones. Then duplicate each circuit into two circuits called x and y. The latches in x are clocked by one phase and the latches in y by the other phase. The outputs of latches in x are used as inputs to the circuit y and vice versa. This generates two-phase circuits, with twice the number of gates and latches. We use the unit fanout delay model; each gate has a delay of 1 plus 0.2 units per each fanout of the gate. The set-up and hold times were set to 0. We constrained the duty cycle of each phase to lie between 0.3 and 0.5 of the total clock cycle. The results (on a DEC5000) are shown in the table below.

name	*size*	*read-in*	*lower*	*time*	*optimal*	*time*		*time ([8])*
	(# nodes/#latches)	(sec.)	*bound*	(sec.)	*clock*	A1(sec.)	A2(sec.)	(sec.)
2shiftreg	28 /6	0.01	7.20	0.01	7.20	0.01	0.01	-
2planet	284 /12	0.22	11.60	0.01	11.60	0.01	0.01	-
2s1423	1314 /148	2.04	145.80	1.15	145.80	0.25	0.23	-
2s5378	5558 /328	2.33	44.40	1.26	44.40	0.14	0.14	-
2s9234	11194 /456	7.00	108.40	0.19	108.40	0.22	0.23	-
2s13207	16054 /1338	8.40	120.80	102.89	120.80	0.40	0.41	-
2s35932	32706 /3456	12.14	77.00	5.75	77.00	0.73	0.72	-
2s38584	38814 /2904	28.79	146.80	19.87	146.80	2.29	2.29	4.1
2s38417	44794 /3272	59.90	84.40	858.46	84.40	3.34	3.49	6.0

Table 1: Table of Results for simple delay model

The second column gives the number of nodes and the third column gives the number of latches in the network. The time taken for constructing the circuit graph is shown in column 3. Column 4 and 5 give the lower bound on the clock cycle and the time required to compute it using the algorithm in [8]. Once this lower bound is computed, in order to perform the optimization of the phases, the constraints S2 of Section 3 need to be solved. This can be done by a linear program solver, as in [8], or by the algorithms proposed in Section 5 and 6. Column 6 gives the optimal clock cycle values. The time required for computation using the algorithm given in the previous sections is shown in columns 7 and 8. A1 is the simplified algorithm in section 5 and A2 is the general algorithm in section

6. The last column gives the time reported by Szymanski [8] to compute the optimal clock for the same examples, with a different delay model and clocking scheme using a standard linear programming package.

Table 2 gives the results for the same circuits mapped into an industrial sequential library. This library has realistic gate delays and set-up/hold times for the memory elements. The run times for the clock schedule optimization algorithms are similar to

name	*size*	*read-in*	*lower*	*time*	*optimal*	*time*	
	(#gates/#latches)	(sec.)	*bound*	(sec.)	*clock*	A1(sec.)	A2(sec.)
2shiftreg	10/6	0.01	5.60	0.01	9.90	0.01	0.01
2planet	574/12	0.62	21.00	0.02	21.00	0.01	0.01
2s1423	852/148	1.92	53.70	1.17	53.70	0.16	0.17
2s5378	2034/326	2.07	14.30	1.28	17.50	0.12	0.11
2s9234	3375/392	3.31	22.30	2.23	25.20	0.24	0.24
2s13207	4954/978	3.76	27.20	4.52	29.70	0.32	0.31
2s35932	16160/3070	17.10	18.40	5.01	18.40	0.66	0.63
2s38584	18780/2900	35.93	34.40	3.12	34.40	2.46	2.31
2s38417	23116/3456	31.67	28.70	2642.62	28.70	2.56	2.75

Table 2: Table of results for "Mapped" circuits

those in Table 1. It was found that a negative cycle (if one exists) is discovered early during the Floyd-Warshall iterations, as a result in most cases the Floyd-Warshall is never completed for lower (*i.e.* infeasible) clock cycles.

9 Conclusion

In this paper we use the circuit model and the clock model proposed by Sakallah *et al.*. The clocking constraints and the technique for eliminating redundant constraints are due to Szymanski. Sakallah and Szymanski used standard linear programming to solve the resulting problem. We have shown that the resulting LP has a special structure and can be efficiently solved. The restricted version of the algorithm has a polynomial bound on the number of operations required for computing the optimal clock cycle. The complexity of the algorithm is $O(|V|l^2 \log |V|)$ (where l is the number of clock phases and $|V|$ the number of memory elements). The general algorithm has a worst case exponential complexity like the simplex method, but for our application it seems to typically take only a few iterations (less than 10). Both algorithms use a "shortest path on a graph" algorithm for the core computation. Resulting run-times are reported for large sequential circuits with both, a simplistic delay model and a complex delay model. Run times for

both algorithms are similar for the two algorithms.

Acknowledgements

We thank Dr. T. Szymanski for several comments on the optimal clocking problem and related issues. We also thank L. Lavagno and K. J. Singh for the help in the preparation of this document. We gratefully acknowledge the support of NSF under grant EMC-8419744, and DARPA under grant JFBI90-073.

A Proof of Theorem 1

We shall drop the superscript on K_{ij}^p to simplify the notation. We point out that for the path p,

$$\sum_{k=0}^{N-1} E_{kk+1} = K_{ji}c + e_i - e_j.$$

S1 $\Rightarrow$ S2) Let G be a graph for which S1 has a solution. Let $p \in \mathcal{P}(G)$. Then p must be one of the four types described. Let us examine the set-up constraints.

- LL (or LF) : Since there can be no flip-flops along p, we get

$$\begin{aligned} A_k &\geq R_{k-1} + D_{k-1k} - E_{k-1k} \\ R_{k-1} &\geq A_{k-1} \end{aligned}$$

implying $A_k \geq A_{k-1} + D_{kk-1} - E_{kk-1}$. Summing from $k = 0$ (*i.e.* i) to $k = N$(*i.e.* j) we get

$$A_i \geq R_j + \sum_{k=1}^{N} D_{k-1k} - (e_i + K_{ji}c - e_j). \tag{16}$$

Together with $A_i \leq e_i^L - S$ and $R_j \geq s_j^L$ we get

$$\begin{aligned} e_i^L - S &\geq s_j^L + D_{ji} - K_{ji}c - e_i + e_j \\ &\Downarrow \\ c + e_i &\geq s_j^L + e_j + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_l + e_i &\geq s_j^L + e_j + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_i &\geq (s_j^L - e_l + e_j) + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_i &\geq s_j + D_{ji} - K_{ji}c + S. \end{aligned}$$

This equivalent to constraint 1 of S2.

- FL (or FF): We have

$$\begin{aligned} A_k &\geq R_{k-1} + D_{k-1k} - E_{k-1k} \\ R_{k-1} &\geq A_{k-1} \end{aligned}$$

implying $A_k \geq A_{k-1} + D_{k-1k} - E_{k-1k}$. Summing from $k = 0$ to $k = N$ we get

$$A_i \geq R_j + \sum_{k=1}^{N} D_{k-1k} - (e_i + K_{ji}c - e_j). \tag{17}$$

Together with $A_i \leq e_i^L - S$ and $R_j = e_j^L$ (since j is a flip-flop) we get

$$\begin{aligned} e_i^L - S &\geq e_j^L + D_{ji} - K_{ji}c - e_i + e_j \\ &\Downarrow \\ c + e_i &\geq e_j^L + e_j + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_l + e_i &\geq e_j^L + e_j + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_i &\geq (e_j^L - e_l + e_j) + D_{ji} - K_{ji}c + S \\ &\Downarrow \\ e_i &\geq e_j + D_{ji} - K_{ji}c + S \end{aligned}$$

This is equivalent to constraint 2 of S2.

It remains to prove the hold time inequalities in S2 for each path consisting of one edge e_{ji}.

- LL (or LF): For any edge e_{ji} we know

$$\begin{aligned} a_i &\leq r_j + d_{ji} - E_{ji} \\ r_j &= s_j^L \\ a_i &\geq H \end{aligned}$$

Hence

$$\begin{aligned} H &\leq s_j^L + d_{ji} - (e_i - e_j + K_{ji}c) \\ &\Downarrow \\ H &\leq (s_j^L + e_j) + d_{ji} - e_i - K_{ji}c \\ &\Downarrow \\ e_i &\leq (s_j^L + e_j - e_l) + d_{ji} + e_l - K_{ji}c - H \\ &\Downarrow \\ e_i &\leq s_j + d_{ji} + (1 - K_{ji})c - H. \end{aligned}$$

This is equivalent to constraint 3 of S2.

- FL (or FF): For any edge e_{ji} we know

$$\begin{aligned} a_i &\leq r_j + d_{ji} - E_{ji} \\ r_j &= e_j^L \\ a_i &\geq H \end{aligned}$$

Hence

$$\begin{aligned} H &\leq e_j^L + d_{ji} - (e_i - e_j + K_{ji}c) \\ &\Downarrow \\ H &\leq (e_j^L + e_j) + d_{ji} - e_i - K_{ji}c \\ &\Downarrow \\ e_i &\leq (e_j^L + e_j - e_l) + d_{ji} + e_l - K_{ji}c - H \\ &\Downarrow \\ e_i &\leq e_j + d_{ji} + (1 - K_{ji})c - H \end{aligned}$$

This is equivalent to constraint 4 of S2.

Thus we have shown that S2 is satisfied if S1 is.

S1 $\Leftarrow$ S2) Let the set S2 be consistent. We will show that there exists a solution to S1 by constructing an algorithm to calculate a set of values that satisfy the constraints in S1. The algorithm is given below. The superscript of each variable reflects the iteration number.

$R_i^0 = s_i^L$ `if i is a latch`
$R_i^0 = e_i^L$ `if i is a flip-flop`
$A_i^0 = \max_{j \in FI(i)} (R_j^0 + D_{ji} - E_{ji})$
`do {`
  `foreach memory element i{`
    `if (i is a latch) {`
      $R_i^{k+1} = \max(A^k, s_i^L)$
    `} else { /* i is a flip-flop*/`
      $R_i^{k+1} = e_i^L$
    `}`
    $A_i^{k+1} = \max_{j \in FI(i)} (R_j^{k+1} + D_{ji} - E_{ji})$
  `}`
`} while (not converged)`
`foreach latch i {`

```
        if (i is a latch) {
                r_i = s_i^L
        } else {
                r_i = e_i^L
        }
}
foreach latch i {
        a_i = min_{j∈FI(i)} (r_j + d_ji − E_ji)
}
```

We need to show that the set of sequences $\{R_i^k\}_{k=0}^{\infty}, \{A_i^k\}_{k=0}^{\infty}$ converge for each i and that the limit points of these sequences satisfy the set-up constraints. It is easy to see that the sequences are monotone increasing. Assume that there exists a latch or flip-flop i for which the sequence $\{A_i^k\}_{k=0}^{\infty}$ either does not converge or converges to a value greater than $e_i^L - S$. Then since the sequence is monotone increasing, $\exists N$, such that $A_i^N > e_i^L - S$. Now we will construct a path p as follows. We let $p_N = i$. We define p_{v-1} recursively as the fanin of p_v for which $A_{p_v}^v - R_{p_{v-1}}^v = D_{v-1v} - E_{p_{v-1}p_v}$. The recursion terminates in four cases

- $v = 0$ is a latch for which $R_{p_0}^0 = s_{p_0}^L$ or
- $v = 0$ is a flip-flop for which $R_{p_0}^0 = e_{p_0}^L$ or
- we find a latch $v = V$ such that $R_{p_V}^V = s_{p_V}^L$ or
- we find a flip-flop $v = V$ such that $R_{p_V}^V = e_{p_V}^L$ or

Note that the first case is isomorphic to the third case with $V = 0$ (henceforth called case 1). Similarly the second case is isomorphic to the fourth case with $V = 0$ (henceforth called case 2).

1. case 1 (LL or LF):

$$\begin{aligned}
A_{p_N}^N &> e_i^L - S \\
A_{p_N}^N - R_{p_{N-1}}^N &= D_{p_{N-1}p_N} - E_{p_{N-1}p_N} \\
R_{p_{N-1}}^N &= A_{p_{N-1}}^{N-1} \\
A_{p_{N-1}}^{N-1} - R_{p_{N-1}}^{N-1} &= D_{p_{N-2}p_{N-1}} - E_{p_{N-2}p_{N-1}} \\
R_{p_{N-1}}^{N-1} &= A_{p_{N-2}}^{N-2} \\
&\vdots \\
R_{p_V}^V &= s_{p_V}^L.
\end{aligned}$$

Summing up we get

$$
\begin{aligned}
s^L_{p_V} + \sum_{m=V+1}^{N} (D_{m-1m} - E_{p_{m-1}p_m}) &> e^L_i - S \\
&\Downarrow \\
s^L_j + D_{ji} - e_i + e_j - K_{ji}c &> e^L_i - S, \qquad \text{where } j = p_V \\
&\Downarrow \\
(s^L_j - e_l + e_j) + D_{ji} - K_{ji}c &> (e^L_i - e_l + e_i) - S \\
&\Downarrow \\
s_j + D_{ji} - K_{ji}c + S - L_{ji}c &> e_i
\end{aligned}
$$

contradicting an S2 constraint (1 or 2 as the case may be) for path p.

2. case 2 (FL or FF):

$$
\begin{aligned}
A^N_{p_N} &> e^L_i - S \\
A^N_{p_N} - R^N_{p_{N-1}} &= D_{p_{N-1}p_N} - E_{p_{N-1}p_N} \\
R^N_{p_{N-1}} &= A^{N-1}_{p_{N-1}} \\
A^{N-1}_{p_{N-1}} - R^{N-1}_{p_{N-1}} &= D_{p_{N-2}p_{N-1}} - E_{p_{N-2}p_{N-1}} \\
R^{N-1}_{p_{N-1}} &= A^{N-2}_{p_{N-2}} \\
&\vdots \\
R^V_{p_V} &= e^L_{p_V}.
\end{aligned}
$$

Summing up we get

$$
\begin{aligned}
e^L_{p_V} + \sum_{m=V+1}^{N} (D_{m-1m} - E_{p_{m-1}p_m}) &> e^L_i - S \\
&\Downarrow \\
e^L_j + D_{ji} + e_i - e_j + K_{ji}c &> e^L_i - S, \qquad \text{where } j = p_V \\
&\Downarrow \\
(e^L_j - e_l + e_j) + D_{ji} - K_{ji}c &> (e^L_i - e_l + e_i) - S \\
&\Downarrow \\
e_j + D_{ji} - K_{ji}c + S &> e_i
\end{aligned}
$$

contradicting an S2 constraint (3 or 4 as the case may be) for path p.

Hence the sequences $\{A_i^k\}_{k=0}^{\infty}$ converge to legal values for each i. This also implies that the sequences $\{R_i^k\}_{k=0}^{\infty}$ converge to legal values for each i. Let the limit point of the sequences be denoted by A_i and R_i for each latch/flip-flop i.

We now prove that hold time constraints are satisfied. For sake of contradiction assume there is a latch or flip-flop i for which the hold time is violated.

$$\begin{aligned} a_i &< H \\ r_j + d_{ji} - E_{ji} &< H \qquad \text{for some } j \in FI(i). \end{aligned}$$

There arise two cases depending on whether j is a flip-flop or a latch.

1. j is a latch: $r_j = s_j^L$

$$\begin{aligned} s_j^L + d_{ji} - e_i + e_j - K_{ji}c &< H \\ &\Downarrow \\ (s_j^L + e_j - e_l) + d_{ji} + e_l - e_i - K_{ji}c &< H \\ &\Downarrow \\ s_j + d_{ji} + c - K_{ji}c - H - L_{ji}c &< e_i \end{aligned}$$

   This contradicts constraint 5 or 6 in S2.

2. j is a flip-flop: $r_j = e_j^L$

$$\begin{aligned} e_j^L + d_{ji} - e_i + e_j - K_{ji}c &< H \\ &\Downarrow \\ (e_j^L + e_j - e_l) + d_{ji} + e_l - e_i - K_{ji}c &< H \\ &\Downarrow \\ e_j + d_{ji} + c - K_{ji}c - H &< e_i \end{aligned}$$

   contradicting constraint 7 or 8 in S2.

It remains to show that $a_i \leq A_i$ for all i when i is a latch. We have

$$\begin{aligned} a_i &= \min_{j \in FI(i)} (r_j^L + d_{ji} - E_{ji}) \\ r_j^L &\leq R_j^L \\ a_i &\leq \min_{j \in FI(i)} (R_j^L + d_{ji} - E_{ji}) \\ d_{ji} &\leq D_{ji} \quad \text{by definition} \\ a_i &\leq \min_{j \in FI(i)} (R_j^L + D_{ji} - E_{ji}) \\ a_i &\leq \max_{j \in FI(i)} (R_j^L + D_{ji} - E_{ji}) \\ a_i &\leq A_i. \end{aligned}$$

□

This proof assumes that $e_j \geq s_j$. Note that in case $e_j < s_j$, there is an additional term of $-c$ on the right hand sides of all the equations that use s_j. The proof is analogous to the one above.

The next important thing to note is a bound on the number of iterations needed for convergence of the iterative scheme described above.

Lemma 4: The iterative scheme will converge in at most $|V| - 1$ iterations.

Proof: It is easy to see that the short path constraints require $|E|$ relaxations (each edge once). The convergence of the long path constraints can be seen by showing that the problem is related to finding the longest path on a graph with non-positive cycles. Weigh each edge $e_{ji} \in E$ with $D_{ji} - E_{ji}$. Each vertex i (memory element) $\in V$ has two variables, A_i, R_i. Initialize all R_i as in the algorithm. Now carry out a relaxation of all the edges, $(|V| - 1)$ times. This ensures every vertex-disjoint path has been considered as a path in the relaxation. Any cycle can be represented as a path $p : i \rightarrow i$. We know $c \geq \frac{D_p}{K_p}$ (see equation 12), where D_p is the sum of the delays along p and K_p is the sum of the E_e for each e lying on p. As a result traversing a cycle can only decrease the arrival times. Consequently we need at most $|V| - 1$ iterations. □

References

[1] M. R. Dagenais and N. C. Rumin. Automatic Determination of Optimal Clocking Parameters in MOS VLSI Circuits. In *Advanced Research in VLSI:Proc. of the 5th MIT Conference*, pages 19–33, 1988.

[2] R. B. Hitchcock. Timing Verification and Timing Analysis Program. In *Proceedings of the Design Automation Conference*. IEEE/ACM, 1982.

[3] A. Ishii and C. E. Leiserson. A Timing Analysis of Level-Clocked Circuitry. In *Advanced Research in VLSI: Proc. of the 7th MIT Conference*, 1990.

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

[5] J. K. Ousterhout. A Switch-Level Timing Verifier for Digital MOS VLSI. *IEEE Transactions on Computer-Aided Design*, CAD-4(3):336–349, July 1985.

[6] K. Sakallah, T. N. Mudge, and O. A. Olukotun. Check T_c and min T_c: Timing Verification and Optimal Clocking of Synchronous Digital Circuits. In *Proceedings of the International Conference on Computer-Aided Design*, pages 552–555. IEEE, 1990.

[7] T. G. Szymanski. LEADOUT: A Static Timing Analyzer for MOS Circuits. In *Proceedings of the International Conference on Computer-Aided Design*, pages 130–133. IEEE, 1986.

[8] T. G. Szymanski. Computing Optimal Clock Schedules. In *Proceedings of the Design Automation Conference*, pages 399–404. IEEE/ACM, 1992.

[9] S. H. Unger and C. J. Tan. Clocking Schemes for High-Speed Digital Systems. *IEEE Transactions on Computers*, C-35(10):880–895, October 1986.

[10] D. Wallace and C. H. Sequin. ATV: An Abstract Timing Verifier. In *Proceedings of the Design Automation Conference*, pages 154–159. IEEE/ACM, 1988.

[11] N. Weiner and A. Sangiovanni-Vincentelli. Timing Analysis in a Logic Synthesis Environment. In *Proceedings of the Design Automation Conference*, pages 655–661. IEEE/ACM, 1989.

[12] R. Zahir. *Controller Synthesis for Application Specific Integrated Circuits*. Hartung–Gorre Verlag, Konstanz, Germany, 1991.