Check for updates

SEQUENTIAL CIRCUIT DELAY OPTIMIZATION USING GLOBAL PATH DELAYS

Srimat T. Chakradhar
Sujit Dey
Miodrag Potkonjak
Steven G. Rothweiler
Computers and Communications Research Labs, NEC USA, Princeton, NJ 08540

ABSTRACT: We propose a novel sequential delay optimization technique based on network flow methods that simultaneously exploits delays on all paths in the circuit. We view the sequential circuit as an interconnection of path segments with pre-specified delays. Path segments are bounded by flip-flops, primary inputs or primary outputs. Recognizing that a delay optimizer can satisfy certain delay constraints more easily than others, we first propose a measure of difficulty for the delay optimizer. Our measure is based on explicit path delays to be satisfied by the delay optimizer. Also, our measure induces a partial order on the set of possible delay constraints. We then compute a set of delay constraints that is optimal with respect to our measure. The delay constraint set is optimal in the sense that it is the easiest constraint that can be specified to the delay optimizer. We formulate the delay constraint calculation problem as a minimum cost network flow problem. If the delay optimizer satisfies the optimal delay constraint set, then the resynthesized circuit may have several paths exceeding the desired clock period. However, we show that the resynthesized circuit can always be retimed to achieve the desired clock period. Experimental results on MCNC synthesis benchmarks show that our method improves the performance of circuits beyond what is achievable using optimal retiming and conventional combinational logic synthesis.

1. INTRODUCTION

Consider a digital circuit S that is specified as an interconnection of combinational logic gates and clocked flip-flops. We assume that all flip-flops are driven by a *single* clock (i.e., single-phase circuits) and the latching is always positive (or always negative) edge triggered. We assume that the clock has a period of ϕ seconds and every gate has a unit propagation delay. Retiming [1] attempts to reduce the clock period of S to $\phi - \epsilon$ ($\epsilon > 0$) by moving the latches in the circuit. The behavior of the retimed circuit S^R is identical to the behavior of S for all input sequences.

If the desired clock period ϕ cannot be achieved by retiming, then a combination of combinational logic resynthesis and retiming can be used to achieve the desired clock period [2, 3, 4]. A *combinational delay optimizer* can be used to resynthesize the combinational logic of the sequential circuit. The delay optimizer attempts to satisfy delay constraints specified as the arrival and required times of the primary inputs and primary outputs, respectively, of the combinational circuit. A simple and natural specification of delay constraints is to assign an arrival time of 0 to all primary inputs and a required time of ϕ to all primary outputs. However, in many cases it may be impossible to resynthesize the circuit to meet this delay constraint [4]. A recent technique [4] enables retiming by using combinational logic transformations. They use forward movement of latches to derive a set of arrival and required times for the inputs and outputs of the combinational logic of the sequential circuit. However, as we show in Section 2, these delay constraints may be unduly restrictive. This is because they are computed based on local information like slacks at latches.

In this paper, we propose a sequential delay optimization technique that simultaneously exploits delays on all paths in the circuit. Recognizing that a delay optimizer can satisfy certain delay constraints more easily than others, we first propose a measure of difficulty for the delay optimizer. We then derive a delay constraint set that is optimal in the sense that it is the easiest constraint that can be specified to the delay optimizer. The optimal delay constraint set is computed by viewing the sequential circuit as an interconnection of path segments with pre-specified delays. Path segments are bounded by flip-flops, primary inputs or primary outputs. We simultaneously consider delays on all path segments and formulate the delay constraint calculation problem as a minimum cost network flow problem. The optimal solution to the flow problem corresponds to an optimal delay constraint set. If the delay optimizer satisfies the optimal delay constraint set, then the resynthesized circuit may have several paths exceeding the desired clock period. However, we show that the resynthesized circuit can always be retimed to achieve the desired clock period. The path segment view of sequential circuits also yields a new retiming technique for unit delay circuits. This technique is presented elsewhere [5].

2. DELAY CONSTRAINT COMPUTATION

Consider a sequential circuit S. Let $L = \{l_1 \ldots l_k\}$ be the set of flip-flops, primary inputs and outputs of S. The pri-

30th ACM/IEEE Design Automation Conference®
Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. ©1993 ACM 0-89791-577-1/93/0006-0483 1.50

mary inputs or latch outputs of S are the primary inputs of its combinational logic. Also, the primary outputs or latch inputs of S are the primary outputs of its combinational logic. In the sequel, we will refer to the primary inputs and primary outputs of the combinational logic as inputs and outputs, respectively. A combinational delay optimizer attempts to satisfy pre-specified maximum tolerable path delays between inputs and outputs of the combinational logic. We refer to the maximum tolerable path delay between a given input and output of the combinational logic as a *delay constraint.*

Different input and output pairs can have different delay constraints. These delay constraints are usually specified as the arrival and required times of the inputs and outputs, respectively, of the combinational logic. We use the following notation to represent the arrival and required times of inputs and outputs, respectively, of the combinational logic. If l_i is a latch, its output is an input to the combinational logic. We denote the arrival time of the latch output by x_i^a. Similarly, the input to latch l_i is also an output of the combinational logic and the required time of the latch input signal is denoted by x_i^r. If l_i is a primary input of S, it is also an input of the combinational logic. The arrival time of this input is denoted by x_i^a. Note that in this case, x_i^r is not defined. Similarly, if l_i is a primary output of S, then it is also an output of the combinational logic. The required time of this output is denoted by x_i^r. Note that in this case, x_i^a is not defined. In the absence of external interface constraints, we assume that the arrival time for all primary inputs is 0 and the required time for all primary outputs is the desired clock period. If external timing constraints are specified, they can be easily incorporated into our delay computation framework.

Forward movements of latches can be used to obtain a set of delay constraints [4]. If latch l_i has an input slack s_i, then they assign the latch output signal an arrival time of $x_i^a = -s_i$. This is possible since the latch can be moved forward by s_i without making any paths terminating at l_i to become critical. However, moving the latch forward by s_i implies that the latch input signal must arrive s_i units of time earlier than the default required time of $\phi - \epsilon$ for all outputs of the combinational logic. Therefore, the new required time for the latch input signal is $x_i^r = \phi - \epsilon - s_i$. The delay optimizer resynthesizes the combinational logic under these delay constraints. However, there are cases when the delay optimizer will fail to meet the delay specifications. Again, there may not exist a combinational logic implementation that satisfies the delay requirements. For example, consider the circuit in Figure 1. The clock period of the

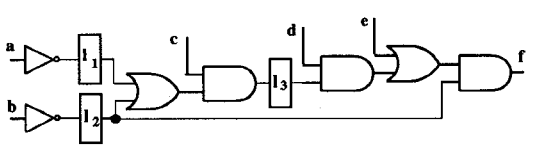

FIGURE 1: An example sequential circuit.

circuit is $\phi = 3$ and this cannot be reduced any further by retiming. This is because there is a combinational path between primary input d and the primary output f that has a delay of 3. Also, combinational delay optimization cannot reduce the delay of the circuit any further. If the desired clock period is 2, latches l_1 and l_2 have an input slack of 1. However, latch l_3 has no input slack. Therefore, $x_1^a = x_2^a = -1$ and $x_3^a = 0$. The arrival time of all primary inputs is 0 and the required time of the primary output f is 2. The required times of the latch input signals of l_1, l_2 and l_3 are $x_1^r = x_2^r = 1$, $x_3^r = 2$. It is impossible to resynthesize the combinational logic to meet these delay constraints since there does not exist an implementation for f that meets the delay constraints. However, as we show later in Section 6, it is possible to compute an *easier* set of delay constraints that can be satisfied by the delay optimizer. A clock period of 2 can be achieved by combinational resynthesis and subsequent retiming. The above example clearly reveals limitations of computing delay constraints based on local information like the input slack of latches.

4. A MEASURE OF DIFFICULTY

A combinational delay optimizer can satisfy certain delay constraints more easily than others. For example, a delay constraint set that requires all path delays to be less than $\phi - \epsilon$ is more stringent than a delay constraint set that requires most paths to have a delay less than or equal to $\phi - \epsilon$ but allows some paths to have a delay more than $\phi - \epsilon$. This is because the delay optimizer may be able to resynthesize the logic to satisfy the latter constraint set but it may fail to satisfy the former constraint set. Also, if the delay optimizer satisfies the former constraint set, it automatically satisfies the latter constraint set.

We propose the following measure of difficulty for combinational delay optimization. Given only structural descriptions of circuits, we use path lengths in the combinational logic to obtain a measure of difficulty. If functional information about the circuit or its internal signals is available, it is possible to incorporate this information into our measure. Let D_1 and D_2 be two delay constraint sets on paths in the combinational logic. Let p be any path in the combinational logic. If the maximum allowable path delay on any path p in constraint set D_1 is always greater than or equal to the corresponding allowable path delay on p in set D_2, then we define $D_1 \leq D_2$. Note that our definition induces a partial order on the delay constraints on paths in the combinational logic. Constraint D_1 is less stringent than D_2 because the delay optimizer may be able to satisfy D_1 but it may fail to satisfy D_2. Also, D_1 is automatically satisfied whenever D_2 is satisfied.

Let D_2 be the set of actual maximum path delays between any input and output pair of the combinational logic. If the combinational logic has m inputs and n outputs, then D_2 can have at most $m \times n$ elements. Since the clock period of S is more than the desired clock period $\phi - \epsilon$, the delay on some paths in the combinational logic exceeds $\phi - \epsilon$. Paths with delays exceeding the desired clock period are called *long* paths and paths with delays less than the desired clock period are called *short* paths. We simultaneously consider

delays on all path segments to obtain a delay constraint D_1 that satisfies the following two conditions:

1. $D_1 \leq D_2$
2. D_1 is the greatest lower bound for D_2. Therefore, there is no delay constraint D_3 so that $D_3 \leq D_1 \leq D_2$.

In a sense, the constraint D_1 is the easiest constraint that can be specified to the delay optimizer. Note that if the resynthesized logic meets the delay constraint D_1, there may be path segments with delays exceeding the desired clock period. However, as we show in Section 7, it is always possible to retime the resynthesized circuit to achieve the desired clock period $\phi - \epsilon$.

5. OPTIMAL DELAY CONSTRAINT SET

The arrival and required times of the inputs and outputs, respectively, of the combinational logic are computed by simultaneously considering all path segments of the sequential circuit. Let the default arrival time of all inputs of the combinational logic be 0. Also, let $\phi - \epsilon$ be the default required time of all outputs of the combinational logic. Note that the primary inputs and outputs of the sequential circuit S assume the default values in any optimal delay constraint set. We specify the arrival times of all inputs of the combinational logic with respect to the primary inputs of S. Similarly, we specify the required times of all outputs of the combinational logic with respect to the required time of the primary outputs of S. The arrival time of the output signal of a latch and the required time of the latch input signal are related as follows. Consider a latch l_i. If the arrival time of the latch output signal is advanced by x_i^a (i.e., this signal arrives x_i^a units of time ahead of the primary inputs of S), then the latch input signal's required time is also advanced by the same amount. Therefore, the input signal of the latch is required to be ready x_i^a units of time ahead of the primary outputs of S. Let x_i be the number of time units by which the output signal of latch l_i is advanced as compared to the primary inputs of S. If x_i is negative, then the output signal of latch l_i arrives $-x_i$ units of time after the primary inputs of S. Also, let x_0 denote the change in the arrival and required times of primary inputs and primary outputs of S.

We formulate the optimization problem by separately considering short and long path segments.

Short Paths: Let p be the maximum delay from latch l_i to l_j. Since we are considering a short path segment, $p \leq \phi - \epsilon$. If we assume that the output signal of latch l_i arrives at the same time as the primary inputs of S, then the input signal of latch l_j is ready before the default required time of $\phi - \epsilon$. Let the input signal of latch l_j arrive x_j units of time before its default required time. This implies that the output signal of latch l_j is ready x_j units of time before the primary inputs of S. Therefore, an additional delay of x_j units of time can be tolerated on all path segments originating from latch l_j. The delay optimizer can resynthesize path segments originating from latch l_j so that their delay does not exceed $\phi - \epsilon + x_j$ rather than the default value of $\phi - \epsilon$. Assuming that the delay optimizer is able to resynthesize these path segments to meet the delay constraint of $\phi - \epsilon + x_j$, some of the resynthesized path segments may have a delay exceeding the desired clock period of $\phi - \epsilon$. However, the delay on these resynthesized path segments can be reduced by moving latch l_j forward by at most x_j units of time during the retiming phase.

A similar argument applies to path segments terminating at latch l_i. If we assume that the input signal of latch l_j arrives at the same time as the primary outputs of S, then the output signal of latch l_i can arrive after the primary input signals have arrived. This is because the path segment between l_i and l_j is short. Let x_i be the number of time units the output signal of latch l_i can arrive after the primary input signals of S have arrived. This implies that the input signal of latch l_i can be ready x_i units of time after the primary outputs of S. Therefore, an additional delay of x_i units of time can be tolerated on all path segments terminating at latch l_i. The delay optimizer can resynthesize path segments terminating at latch l_i so that their delay does not exceed $\phi - \epsilon + x_i$ rather than the default value of $\phi - \epsilon$. Again, assuming that the delay optimizer is able to resynthesize these path segments to meet the delay constraint of $\phi - \epsilon + x_j$, some of the resynthesized path segments may have a delay exceeding the desired clock period of $\phi - \epsilon$. However, the delay on these resynthesized path segments can be reduced by moving latch l_i backward by at most x_i units of time during the retiming phase.

We now analyze the more general case where the arrival times of output signals of both latches are advanced. Let x_i and x_j be the amounts by which the output signals of latches l_i and l_j, respectively, are advanced. If latch output signals are assigned their default arrival times, then the delay optimizer must resynthesize the path segment between l_i and l_j so that the delay does not exceed $\phi - \epsilon$. If we advance only the output signal of latch l_i, then a delay of $\phi - \epsilon + x_i$ can be tolerated between latches l_i and l_j. However, if we advance only the output signal of latch l_j, then a delay of only $\phi - \epsilon - x_j$ can be tolerated between the two latches. Note that if $\phi - \epsilon - x_j \geq p$, then the delay optimizer does not have to resynthesize the path segment between the latches since the delay constraint is already satisfied by the current implementation. If output signals of both latches are advanced, then the delay optimizer must resynthesize the path segment between the two latches so that the delay does not exceed $\phi - \epsilon - \Delta p$. Here, $\Delta p = x_j - x_i$ is the net decrease in the tolerable delay between the two latches as compared to the default tolerable delay of $\phi - \epsilon$. If $\phi - \epsilon - \Delta p$ becomes less than the original delay of p, then it may be impossible to resynthesize the logic to achieve this delay bound. Therefore, we require that $\phi - \epsilon - \Delta p \geq p$.

Long Paths: Let p be the maximum delay from latch l_i to l_j. Clearly, $p > \phi - \epsilon$. If output signals of the two latches are assigned their default arrival times, then the delay optimizer must resynthesize the path segment between l_i and l_j to reduce the delay from p to $\phi - \epsilon$. If we advance only the output signal of latch l_i, then a delay of $\phi - \epsilon + x_i$ can be tolerated between latches l_i and l_j and the delay optimizer will be required to reduce the delay of this path segment from p to $\phi - \epsilon + x_i$ rather than $\phi - \epsilon$. Assuming that the

delay optimizer is able to resynthesize these path segments to meet the delay constraint of $\phi - \epsilon + x_i$, some of the resynthesized path segments may have a delay exceeding the desired clock period of $\phi - \epsilon$. However, the delay on these resynthesized path segments can be reduced by moving latch l_i forward by at most x_i units of time during the retiming phase.

If we advance only the output signal of latch l_j, then only a delay of $\phi - \epsilon - x_j$ can be tolerated between the two latches. The delay optimizer will have to reduce the delay of this path segment from p to $\phi - \epsilon - x_j$ which may be more difficult to achieve than the original goal of $\phi - \epsilon$. If output signals of both latches are advanced, then the delay optimizer must resynthesize the path segment between the two latches to reduce the delay from p to $\phi - \epsilon - \Delta p$. Clearly, we require that $\Delta p \leq 0$. Otherwise, the delay optimizer will have to reduce the delay from p to a quantity that is lower than $\phi - \epsilon$ and this may be impossible to achieve.

The smaller the value of Δp, the less stringent is the delay constraint for the delay optimizer. However, Δp need not decrease beyond $\phi - \epsilon - p$. This is because at this value of Δp, the tolerable delay on the path segment is equal to p and this delay constraint is already satisfied by the current implementation. Therefore, the delay optimizer does not have to resynthesize the path segment. The fact that Δp need not decrease beyond $\phi - \epsilon - p$ can be captured in an optimization framework as follows:

$$\begin{aligned} &Minimize && \epsilon_{ij} \\ &subject\ to && \Delta p - \epsilon_{ij} \leq \phi - \epsilon - p \\ & && \epsilon_{ij} \geq 0 \end{aligned}$$

Objective function: We construct an objective function that is heavily biased towards increasing the tolerable delay on long path segments so that the tolerable delay is equal to the delay in the current implementation. This amounts to minimizing $\sum \epsilon_{ij}$ for all long path segments. A secondary goal is to increase the tolerable delays on all path segments.

Let P be the set of all path segments. Also, let P_1 and P_2 be the set of short and long path segments, respectively. We denote the path segment from l_i to l_j as $l_i \Rightarrow l_j$. Let d_{ij} be the delay of this segment. The optimization problem to obtain the optimal delay constraint can be stated as follows:

$$Maximize \qquad -\alpha \sum_{l_i \Rightarrow l_j \in P_2} \epsilon_{ij} + \beta \sum_{l_i \Rightarrow l_j \in P} x_i - x_j$$

$$\begin{aligned} subject\ to \quad & l_i \Rightarrow l_j \in P_1 : && x_j - x_i \leq \phi - \epsilon - d_{ij} \\ & l_i \Rightarrow l_j \in P_2 : && x_j - x_i \leq 0 \\ & && x_j - x_i - \epsilon_{ij} \leq \phi - \epsilon - d_{ij} \\ & && \epsilon_{ij} \geq 0 \end{aligned}$$

Here, α is significantly larger than β. The optimization is performed under the following constraints:

- Tolerable delay on a short path segment is greater than or equal to the actual delay of the path segment.
- Tolerable delay on long paths is greater than or equal to the desired clock period.

A solution of the optimization problem may have have $x_0 \geq 0$. Therefore, the arrival time for the output signal of latch l_i is given by $x_i - x_0$.

The above optimization problem is the dual [6] of the minimum cost flow problem. We will refer to the above optimization problem as the *dual* problem and the minimum flow cost problem as the *primal* problem. The network for the flow problem consists of a vertex for each variable x_i in the dual. If the dual has a constraint $x_j - x_i \leq c$, then the network has an arc from j to i. Furthermore, the cost of unit flow over this arc is equal to c and this arc can carry an arbitrarily large amount of non-negative flow. If the dual has a constraint $x_j - x_i - \epsilon_{ij} \leq c$, then the network has an arc from j to i. The cost of unit flow over this arc is equal to c and the flow on this arc cannot exceed α. The coefficient of x_i in the dual objective function is the net flow out of vertex i in the flow network. If the net flow is positive (negative), then vertex i is a source (sink). If the net flow is 0, then vertex i is a transhipment node of the network and the total flow is conserved.

A useful variation of the above problem is as follows. Among long paths, we may prefer to decrease certain longer paths more than others. Our preference may be dictated by functional information available about the long paths. This can be easily incorporated into the objective function as follows. If the maximum path delay between l_i and l_j is p ($p \geq \phi - \epsilon$), then we include the term $p \times (-\Delta p)$ in the objective function. Another variation would be to require that the arrival times (required times) of any latch output (input) signal be ahead of the primary inputs (outputs). All these variations translate into additional constraints that can be easily added to the basic optimization framework. Many other variations are possible using the above optimization framework.

A systematic procedure to obtain the optimal set of delay constraints is as follows:

1. Construct the *path graph* for circuit S. A path graph $\mathcal{P}$ has a vertex l_i for every latch l_i. Primary inputs and primary outputs of circuit S are represented by a single vertex l_0. If there is some path from latch l_i to l_j in circuit S, graph $\mathcal{P}$ has an arc from vertex l_i to vertex l_j with a weight equal to the maximum path delay from latch l_i to latch l_j. If l_i is a primary input, then there is an arc from l_0 to l_j. Similarly, if l_j is a primary output, then there is an arc from l_i to l_0. Combinational paths between primary inputs and primary outputs are not included in the path graph since the delays on these paths can only be reduced by combinational resynthesis.
2. Classify arcs into short and long arcs. An arc is long (short) if its weight exceeds (is less than) the desired clock period.
3. Formulate inequalities for short and long arcs. There is one inequality for every short arc and three inequalities for every long arc.

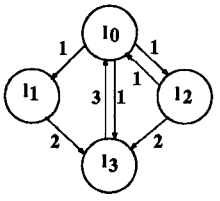

FIGURE 2: Path graph for circuit of Figure 1.

4. Construct objective function.
5. Solve optimization problem using minimum-cost flow algorithm.

Let X_i, $0 \leq i \leq k$ be the optimal arrival time of the output signal of latch l_i. If X_0 is non-zero, then we adjust the arrival time for the latch output signal to be $X_i - X_0$. This translation is done since there is no change in the arrival times of the primary inputs. In the sequel, we assume that this adjustment (if necessary) has been performed and that X_i refers to the adjusted arrival time for the latch output signal. The optimal arrival and required times for the combinational delay optimizer are obtained as follows:

1. Primary inputs are assigned an arrival time of 0. Also, the output of latch l_i is assigned an arrival time of $-X_i$.
2. Primary outputs are assigned a required time equal to the desired clock period $\phi - \epsilon$. All other latch inputs are assigned a required time of $\phi - \epsilon - X_i$.

6. AN EXAMPLE

We illustrate the delay constraint set calculation by an example. Consider the circuit shown in Figure 1. The clock period of the circuit is $\phi = 3$ and this cannot be reduced any further by retiming. This is because there is a combinational path between primary input d and the primary output f that has a delay of 3. Also, combinational delay optimization cannot reduce the delay of the circuit any further. This is because the primary output function f cannot be resynthesized to achieve a clock period of 2. We show that combinational resynthesis using the optimal delay constraint set reduces the clock period to 2. Therefore, the reduction in clock period is $\epsilon = 1$.

The path graph for the circuit in Figure 1 is shown in Figure 2. It has three vertices l_1, l_2 and l_3 corresponding to the three latches in the circuit. Vertex l_0 corresponds to the primary inputs and primary outputs of the circuit. Since there is a path from primary inputs to latch l_1 with a maximum delay of 1, we include the arc $l_0 \Rightarrow l_1$ with a weight of 1 in the path graph. Similarly, paths from latch l_3 to primary outputs are represented by the arc $l_3 \Rightarrow l_0$. The weight of this arc is 3 since the maximum delay on any path from l_3 to a primary output is 3. Other arcs in the path graph can be constructed similarly.

Arc $l_0 \Rightarrow l_3$ is a short arc and the corresponding inequality is $x_3 - x_0 \leq 1$. Similar inequalities are constructed for the remaining five short arcs in the path graph. The path graph has only one long arc $l_3 \Rightarrow l_0$. This arc contributes three inequalities: $x_0 - x_3 \leq 0$, $x_0 - x_3 - \epsilon_{30} \leq -1$ and $\epsilon_{30} \geq 0$. The optimization problem can be formulated directly from the path graph:

$$
\begin{array}{lccc}
\textit{Maximize} & -\alpha\epsilon_{30} + \beta(x_0 + x_2 - 2x_3) & & \\
\textit{subject to} & x_1 - x_0 \leq 1 & & x_2 - x_0 \leq 1 \\
 & x_3 - x_0 \leq 1 & & x_3 - x_1 \leq 0 \\
 & x_3 - x_2 \leq 0 & & x_0 - x_2 \leq 1 \\
\epsilon_{30} \geq 0 & x_0 - x_3 - \epsilon_{30} \leq -1 & & x_0 - x_3 \leq 0
\end{array}
$$

The first six inequalities correspond to the short arcs. The last three inequalities correspond to the long arc $l_3 \Rightarrow l_0$. We solve the optimization problem using a minimum-cost flow algorithm and obtain the solution: $x_0 = 0$, $x_1 = 1$, $x_2 = 1$ and $x_3 = 1$ We assume that $\alpha = 10$ and $\beta = 1$. The arrival times for all primary inputs are 0. The arrival times for outputs of latches l_1, l_2 and l_3 are -1, -1 and -1, respectively. The required time for all primary outputs is the desired clock period 2. Required times for the inputs of latches l_1, l_2 and l_3 are 1, 1 and 1, respectively. We resynthesize the combinational logic under these delay constraints. The resynthesized circuit is shown in Figure 3. The delay optimizer has satisfied all the specified delay constraints. However, note that the resynthesized combinational logic has paths exceeding the desired clock period of 2. However, as shown in the next section, this circuit can always be retimed to achieve the desired clock period. The retimed circuit is shown in Figure 4.

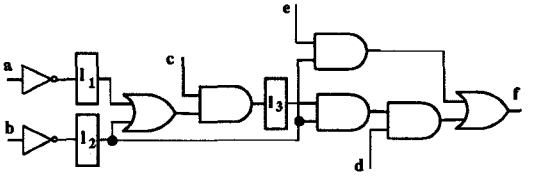

FIGURE 3: Resynthesized circuit.

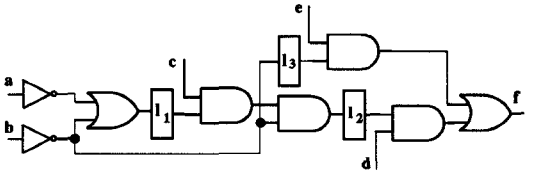

FIGURE 4: Retiming of resynthesized circuit.

7. RESYNTHESIZED CIRCUIT IS RETIMABLE

Theorem 1: *Let S' be the circuit obtained by resynthesizing circuit S using the optimal delay constraint. Circuit S' is always retimable to achieve a clock period of $\phi - \epsilon$.*

Proof: It suffices to show that the resynthesized circuit S' has no critical paths or cycle (See **APPENDIX**).

Absence of critical paths: Consider a path with primary input l_0 and primary output l_{n+1} with n latches labeled

$l_2 \ldots l_{n-1}$. We show that the delay of this path is less than or equal to $(n+1) \times (\phi - \epsilon)$. Let p_{ij} be the delay between l_i and l_j. Therefore, the delay of this path is bounded by $\sum_{i=0}^{i=n} p_{i,i+1}$. Since resynthesis of S guarantees that $p_{i,i+1} \leq (\phi - \epsilon) - (x_{i+1} - x_i)$ and $x_0 = x_{n+1} = 0$, the summation is bounded by $(n+1) \times (\phi - \epsilon)$.

Absence of critical cycles: Consider a cycle in S' with latches $l_1 \ldots l_n$. We show that the delay of this cycle is less than or equal to $n \times (\phi - \epsilon)$. Again, let p_{ij} be the delay between l_i and l_j. Therefore, the delay of this path is bounded by $\sum_{i=1}^{i=n} p_{i,i+1}$. Here, $p_{n,n+1}$ is the delay between latches l_n and l_1. Using $p_{i,i+1} \leq (\phi - \epsilon) - (x_{i+1} - x_i)$, the summation is bounded by $n \times (\phi - \epsilon)$. ∎

8. EXPERIMENTAL RESULTS

We implemented the proposed delay optimization technique in a prototype *C* language program called SDO (sequential delay optimizer). Our implementation consists of three main parts: retiming, delay constraint computation and combinational resynthesis. Retiming and combinational resynthesis in SDO are performed using the unit delay retiming and *speed_up* tools, respectively, that are part of the logic synthesis framework SIS [7]. Delay constraint computation in SDO is performed using a commercial linear programming package called CPLEX [8] that also solves network flow problems.

Table 1 summarizes the experimental results on the MCNC synthesis benchmarks. We transform every circuit into a circuit that consists of only two-input NAND gates by using the *tech_decomp -a 2* program in SIS [7]. The circuit obtained after using *tech_decomp* is the initial circuit for our experiments. Under column *Initial*, we show the area, the number of flip flops and the clock period of the initial circuit. The area of a circuit is the number of literals in the circuit. The number of flip flops in the circuit is indicated under column *Reg.*. The clock period of the circuit is indicated in column ϕ.

For each circuit, we conducted three experiments. For a fair comparison, we used the same retiming and combinational delay optimizer (*speed_up*) for all experiments. In the first experiment, we used the retiming program in SIS to obtain an optimally retimed circuit. The area, the number of flip flops and the clock period of the optimally retimed circuit are shown under the column *Only Retiming*. In the second experiment, we performed optimal retiming as well as combinational delay optimization using the *speed_up* program in SIS with arguments *-d 6 -m unit*. Column *Retiming & speed_up* shows the area, number of flip flops and the clock period of the circuit obtained by using a combination of optimal retiming and *speed_up*. Finally, column SDO shows the area, number of flip flops and the clock period of the circuit obtained by using our delay optimization technique. The delay constraint calculation part of our program took less than one second of CPU time on a Sparc2 workstation for all example circuits.

As an example, the circuit **dk14** initially has a clock period of 14. It has 283 literals and three flip flops. After optimal retiming, the clock period of the circuit reduces to 13. If we use a combination of retiming and *speed_up*, then a clock period of only 12 is achievable. When the optimally retimed circuit is processed by SDO, the clock period reduces from 13 to 8. The optimized circuit has 310 literals and three flip flops. Experimental results clearly indicate that it is possible to reduce the clock period of the circuit beyond what is achievable using optimal retiming by using global path delays in the circuit.

9. CONCLUSION

We have presented a new framework for sequential delay optimization that improves the performance of circuits beyond what is possible by using optimal retiming. The sequential circuit is viewed as an interconnection of weighted path segments and a network flow formulation is used to obtain an optimal set of delay constraints. If desired, our framework can exploit functional information about paths to bias delay optimization. Our formulation also provides a new technique for retiming unit delay synchronous circuits [5]. We are currently considering the initialization and register minimization of circuits produced by SDO.

Acknowledgments: We thank A. Casavant and A. Ishii for several useful discussions.

REFERENCES

[1] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.

[2] G. D. Micheli, "Synchronous Logic Synthesis: Algorithms for Cycle-Time Minimization," *IEEE Transactions on Computer-Aided Design*, vol. 10, pp. 63–73, January 1991.

[3] S. Malik, E. Sentovich, R. Brayton, and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks with Combinational Techniques," *IEEE Transactions on Computer-Aided Design*, vol. 10, pp. 74 – 84, January 1991.

[4] S. Dey, M. Potkonjak, and S. Rothweiler, "Performance Optimization of Sequential Circuits by Eliminating Retiming Bottlenecks," in *IEEE Proceedings of the International Conference on Computer-Aided Design*, November 1992.

[5] S. T. Chakradhar, S. Dey, M. Potkonjak, and S. Rothweiler, "Sequential Circuit Delay Optimization Using Global Path Delays," Tech. Rep. 93-C006-4-5506-3, NEC USA, Princeton, NJ, 1993.

[6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity*. Englewood Cliffs, New Jersey: Prentice Hall, 1982.

[7] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," in *IEEE International Conference on Computer Design*, pp. 328–333, October 1992.

[8] *CPLEX*. CPLEX Optimization, Inc., 930 Tahoe Blvd., Bldg. 802, Incline Village, NV 89451-9436, 1992.

APPENDIX

We show that a critical path or cycle is *necessary* and *sufficient* condition to prevent retiming. We model digital circuit as a directed graph G that has a vertex for every primary input, primary output or combinational logic gate. There is an arc e from vertex u to vertex v (represented as $u \rightarrow^e v$) if gate u is an input to gate v. Furthermore, we associate a *delay* $d_v \geq 0$ with vertex v and a *weight* $w_e \geq 0$ with every arc e. Here, d_v is the propagation delay of gate v and w_e is the number of latches on the arc e. The *augmented* graph H is obtained from the graph G by replacing

TABLE 1: Delay optimization results.

Circuit	Initial			Only Retiming			Retiming & *speed_up*			SDO		
	Area	Reg.	ϕ	Area	Reg.	ϕ	Area	Reg.	ϕ	Area	Reg.	ϕ
ex7	47	2	6	47	2	6	45	2	5	36	2	4
bbtas	53	3	5	53	3	5	49	3	5	45	3	4
bbara	99	3	11	99	3	10	100	3	7	106	4	6
dk512	112	4	16	112	4	16	134	4	11	156	4	10
dk17	174	3	12	174	3	11	171	3	10	206	3	8
opus	172	4	14	172	4	14	213	4	10	298	4	9
dk14	283	3	14	283	3	13	270	3	12	310	3	8
sse	267	4	13	267	4	13	276	4	10	224	6	8
bbsse	267	4	13	267	4	13	276	4	10	224	6	8
s208	161	8	10	161	8	10	179	8	8	161	18	7
s420	343	16	12	343	16	12	355	16	10	351	40	9
s510	438	6	12	438	6	11	441	6	10	451	6	9
sbc	1370	28	15	1370	28	15	1453	28	11	1503	27	10

primary input and primary output vertices in graph G by a single *host* node. All outgoing arcs from primary inputs G now originate from the host in the augmented graph H. Similarly, all incoming arcs to primary outputs in G are now incident to the host. The delays of vertices in G and H are also the same, except that the host vertex is assigned a delay of zero. The arc weights in H are the same as the arc weights in G. However, we we increment the weight on all incoming arcs of the host by 1 and we will refer to such arcs as *latch arcs*. Otherwise, a combinational path in G, will result in a zero weight cycle in H.

It is convenient to introduce a new graph H' that has the same vertices and arcs as H. Vertices in H' have the same delay as the corresponding vertices in H. However, the arc weight w'_e of any arc e in H' is $w'_e = w_e - \frac{1}{\phi-\epsilon}$ except for outgoing arcs of the host that have identical weights in H and H'. We obtain a retiming of H by using the graph H'. A similar construction has been used by Leiserson and Saxe [1]. However, in their construction, weights of outgoing arcs of the host are also decreased by $\frac{1}{\phi-\epsilon}$. If G has path from a primary input to a primary output with delay $\phi - \epsilon$, their construction of H' results in the following problem. The corresponding cycle in H' will have a negative weight of $-1 + 1 - \frac{1}{\phi-\epsilon} = -\frac{1}{\phi-\epsilon}$ and we wrongly conclude that G is not retimable.

Leiserson and Saxe [1] (Theorem 11) have identified necessary and sufficient conditions under which H is retimable. They assume that *all* vertices in H have unit delay. However, in our case, the host node in H has zero propagation delay. Furthermore, we are interested in a specific retiming in which H^R has at least one latch on every latch arc. The retiming proposed by Leiserson and Saxe [1] does not guarantee this. We first show that G is retimable if and only if the augmented graph H can be retimed so that each latch arc has a weight of 1. Given such a retiming of H, we obtain G^R as follows. We delete the host node and remove a latch from every incoming arc to the host node in H^R.

Lemma 1: *The graph H is retimable with every latch arc in H^R having a positive weight if and only if H' does not have a negative weight cycle.*

Proof: Assume that H' has no negative weight cycle. We shall produce a retiming r of H so that the clock period is less than or equal to $\phi - \epsilon$. Let $g(v)$ be the weight of the shortest path from v to v_h, the host vertex in H'. We define the retiming function r as follows: $r(v) = \lceil \frac{1}{\phi-\epsilon} + g(v) \rceil - 1$. We claim that this a legal retiming of H. Using the retiming function r, it can be shown that (1) every arc in the retimed graph has non-negative edge weight and (2) that there is at least one latch on any path with delay greater than $\phi - \epsilon$ [5].

We now show that the proposed retiming guarantees that every latch arc has at least one latch. Consider the latch arc $u \rightarrow^e v_h$. Clearly, $r(v_h) = \lceil \frac{1}{\phi-\epsilon} + g(v_h) \rceil - 1 = 0$ since $g(v_h) = 0$ due to the absence of negative cycles. Also, since there are no negative cycles in H and u has only path to the host vertex, $g(u) = g(v_h) + w(e) - \frac{1}{\phi-\epsilon} = w(e) - \frac{1}{\phi-\epsilon}$ Therefore, $r(u) = w(e) - 1$. Therefore, $r(v_h) - r(u) + w(e) = 1$ and H^R has a latch on every latch arc.

If H' contains a negative weight cycle, a retiming of H is impossible. Consider a cycle p in H' that includes the host and has n arcs. Therefore, the delay of this cycle is $n - 1$. Since the cycle has a negative weight, $w(p) - \frac{n-1}{\phi-\epsilon} < 0$. Here, $w(p)$ is the weight of the corresponding cycle in H. Since the number of latches is less than $\frac{n-1}{\phi-\epsilon}$, therefore, this is a critical cycle. Similarly, consider a cycle p that does not include the host and has n arcs. The delay of this cycle is n. Therefore, $w(p) - \frac{n}{\phi-\epsilon} < 0$ and the cycle is critical. Hence, if H' has a negative cycle, then retiming is impossible. ∎

Theorem 2: *A unit delay synchronous circuit S that has a clock period ϕ can be retimed to achieve a clock period $\phi - \epsilon$ ($\epsilon > 0$) iff S has no critical paths or cycles.*

Proof: If S has a critical path or cycle, then S cannot be retimed [4]. We show that if S is not retimable, then S has a critical path or cycle. Let H be the augmented graph of circuit S. From Lemma 1, H is not retimable *iff* H' has a negative weight cycle. Furthermore, a negative weight cycle in H' corresponds to a critical cycle in H. Critical cycles in H that include the host correspond to critical paths in S. Other critical cycles in H correspond to critical cycles in S. If there exists a retiming of H so that every latch arc has a weight of at least one, then circuit S^R is obtained from H^R by deleting the host node and by removing a latch from every latch arc. ∎