A Method for Correcting the Functionality of a Wire-Pipelined Circuit*

Vidyasagar Nookala Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota
Minneapolis, MN 55455
{vidya,sachin}@ece.umn.edu

ABSTRACT

As across-chip interconnect delays can exceed a clock cycle, wire pipelining becomes essential in high performance designs. Although it allows higher clock frequencies, it may change the microarchitecture altogether because of the arbitrary increase in the latencies of the paths and cycles of the circuit. This paper proposes a method to regain the functionality of a wire-pipelined circuit. In this approach, increased cycle latencies are compensated by slowing down the issue rate of the inputs. Our method finds the optimal value of the slowdown required for a circuit as it directly affects the throughput of the circuit. We also incorporate area minimization in our formulation to minimize the number of extra flip-flops added to the circuit. The formulation is tested on circuits derived from ISCAS benchmarks and the results suggest that wire pipelining increases the overall throughput in most of the cases.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*

General Terms

Performance, Algorithms

Keywords

Wire pipelining, Synchronous design

1. INTRODUCTION

Semiconductor industry trends suggest that the operating frequencies of leading edge integrated circuits approximately double every process generation [1], in tune with the projections of Moore's Law. However, wire delays have become a dominant factor in determining the system performance, which is more evident in deep submicron (DSM) technologies. In particular, the shrinking clock periods have made across-chip communication a performance bottleneck, where some global wires may have delays larger than the intended clock period. The scenario is further aggravated by the fact that die sizes increase by 7% with every process generation [1], resulting in even longer wire lengths, and hence longer wire delays. Even the theoretically best optimizers cannot overcome the criticality of the global interconnects. For instance, even after aggressive optimization, delay of a 2cm global interconnect, a common occurrence in DSM designs, is projected to be 0.67ns in 70nm technology [2], placing an upper bound of about 1.5GHz on the operating frequency, much less than the multigigahertz frequencies projected for that technology. This suggests that multicycle across-chip communication is a necessity to support higher operating frequencies. Several approaches can be used to address the criticality of across-chip interconnects, such as:

- *Adopting a Globally Asynchronous Locally Synchronous (GALS) [3] design methodology:* In this approach, the communication between the synchronous subsystems (or blocks) of a circuit, each of which can have a different clock, is based on a full handshake protocol. Several other works have been proposed based on this approach, such as [4, 5] to cite a few. Carloni *et al.,* proposed a latency insensitive design in [6]. However, the overhead for the asynchronous interface may affect both the performance and the area of the design.
- *Providing a slower clock for the flip-flops latching signals from global wires:* Each of the signals from the global wires whose delay is greater than the system clock cycle are latched by the flip-flops clocked by the new, slower clock network. However, this approach adds new complications in the form of routing the extra clock network and synchronization between the clock domains. Moreover, since the slower clock must consider the worst case across-chip wire delay, latching signals from wires whose delay is considerably smaller than the clock cycle degrades the throughput of the circuit.
- *Pipelining the global wires of the circuit:* The delay of an interconnect is distributed over several clock cycles by inserting flip-flops, which allows a fully synchronous operation at higher clock frequencies. A retiming [7] based method for wire pipelining is proposed in [8]. However, since the latencies of the cycles and input-output paths of the circuit remain unchanged in this approach, there is a lower bound on the achievable clock cycle time. In contrast, the techniques proposed in [9, 10] insert flip-flops to pipeline an interconnect to enable higher clock frequencies, in conjunction with repeater insertion. Although pipelining the wires of a circuit using [9, 10] permits higher operating frequencies, the resultant wire-pipelined circuit may be functionally different from the initial circuit. This happens because wire pipelining can arbitrarily increase the latencies of paths and cycles of the circuit due to the insertion of extra flip-flops.

This paper will focus on the aftereffects of wire pipelining. Given a circuit and a wire pipelined version of the circuit, which may be

*This work was supported in part by the NSF under award CCR-0205227, and by the SRC under award 2003-TJ-1092.

DAC 2004, June 7–11, 2004, San Diego, California, USA.

functionally incorrect, we formulate a method to regain the correctness of the wire pipelined circuit.

2. PROBLEM DESCRIPTION

A typical design flow may proceed as follows. After the blocks and modules of the circuit are designed subject to a clock frequency, a block-level placement of the circuit is performed. Wire pipelining is then carried out on the global wires of the circuit, sometimes concurrently with routing [10], or sometimes after routing is done [9], and this may insert flip-flops on a wire if the delay of the wire exceeds a clock cycle. After the wires of a circuit are pipelined, the following two problems must be resolved:

- Increase in the latencies of the cycles of the circuit.
- Nonuniform increase in the latencies of different paths to a block from the inputs of the circuit.

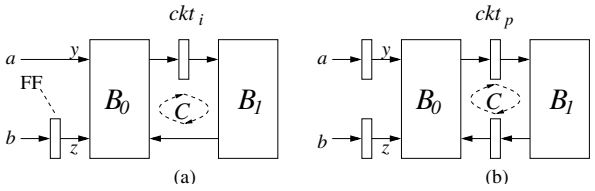

Figure 1: A circuit with two inputs a and b. Signals y and z are the input ports of the block B_0. (a) The circuit before pipelining its wires (ckt_i). (b) The circuit after pipelining its wires (ckt_p).

In this paper, we assume that all the flip-flops are edge-triggered. Consider Figure 1, which depicts a circuit comprising two combinational logic blocks B_0 and B_1, before and after pipelining the wires of the circuit. The two scenarios are labeled ckt_i and ckt_p, as shown in Figures 1(a) and 1(b), respectively. The insertion of an extra flip-flop on the cycle C increases its latency to 2 in ckt_p from 1 in ckt_i. Hence, the output of each block of C propagates back to itself after 1 clock cycle in ckt_i, whereas it takes an extra clock cycle in ckt_p, thus altering the original functionality of the cycle. Moreover, with the insertion of an extra flip-flop between a and y, the inputs a and b reach y and z, respectively, after an equal number of clock cycles in ckt_p, which is not the case in ckt_i. Hence, ckt_i and ckt_p are not functionally equivalent.

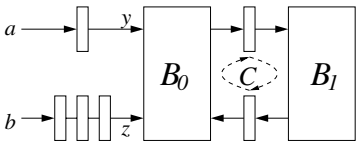

Figure 2: A solution to the problem shown in Figure 1. We refer to this circuit as ckt_f.

Wire pipelining can therefore result in a totally different microarchitecture. This is not the desired result and therefore, must be corrected, and this paper proposes a method for doing so. The solution lies in ensuring that every block receives its inputs at the correct clock cycle. For increased cycle latencies, we use an approach similar to the *c-slow* concept mentioned in [7]. The idea is to *slowdown* the input issue rate[1] of the circuit by some factor ρ, i.e., inputs are allowed to change only every ρ^{th} clock cycle. The issue rate of the initial circuit ckt_i is assumed to be 1.

For instance, the cycle C of ckt_p will be functionally equivalent to the cycle C of ckt_i, if the inputs a and b are permitted to change only every other clock cycle in ckt_p. As a result, ckt_p computes its outputs only every 2 clock cycles, which indicates a reduction in the throughput of the circuit. Moreover, the latency difference between any two paths to a block from the inputs of a circuit must also be maintained in its wire-pipelined version. Going by this argument, since the latency difference between the paths $b \rightarrow z$ and $a \rightarrow y$ is 1 in ckt_i, and 0 in ckt_p, one extra flip-flop must be inserted on the path $b \rightarrow z$ in ckt_p to make it functionally equivalent to ckt_i. However, the slowdown has implications on the path latencies of a wire-pipelined circuit. For example, the latency difference of the paths $a \rightarrow y$ and $b \rightarrow z$ in ckt_i must be amplified by a factor of $\rho = 2$ in ckt_p, since it receives its inputs only every 2 clock cycles. Therefore, 2 extra flip-flops must be inserted on the path $b \rightarrow z$ in ckt_p, as shown in Figure 2.

Our work finds the minimal value of slowdown required for a circuit as this directly affects its throughput and also minimizes the increase in area due to the insertion of extra flip-flops.

[1]The issue rate is defined as the number of clock cycles between successive input changes. An issue rate of 1 indicates that the inputs can change every clock cycle.

3. PRELIMINARIES

In the example in section 2, it was assumed that all blocks were purely combinational. In general, a circuit may have sequential as well as combinational blocks, i.e., the blocks may have internal flip-flops and/or cycles. The existence of cycles in a circuit may require that extra flip-flops be inserted within a sequential block of the circuit. For instance, consider a scenario where there are two paths from an input of a sequential block to one of its outputs. If the two paths have different latencies, and if the circuit requires a slowdown $\rho > 1$, then the solution may require that the difference of latencies be increased by a factor of ρ. Therefore, all of the wires of the block must be considered for the insertion of extra flip-flops. However, in most cases, the blocks are internally undefined blocks at an early stage of design, or IP cores, and therefore, arbitrary insertion of extra flip-flops on the wires within the blocks is not desirable. To avoid this, we use an abstract model for a sequential block that decomposes it into a set of combinational sub-blocks, interconnected by wires having flip-flops. This ensures that for any sequential block, only those interconnections that have flip-flops on them are considered for insertion of extra flip-flops. Figure 3 shows a sequential block and the abstract model of the block. The block is modeled as two combinational sub-blocks, S_1 and S_2, with flip-flops on the interconnections between them.

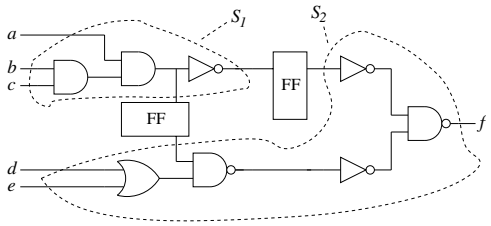

Figure 3: A sequential block and its abstracted model.

For a general circuit, we will consider three scenarios: the *initial circuit*, a *wire-pipelined* version of the initial circuit, and a *corrected wire-pipelined* version of the initial circuit. Flip-flops and repeaters apart, each of the three circuits comprises of the same placed and routed combinational block level or sub-block level netlist. Each net of the circuits is a routed tree that connects the output of a block/sub-block (source) to the inputs of other blocks/gates (sinks) through branch points such as Steiner points [11]. We use three edge weighted graphs to model the three scenarios. The graphs have the same vertex and edge sets, represented as V and E, respectively. The vertex set V of the graphs models the blocks/sub-blocks, the inputs, the outputs and the branch points of the circuit. The set E is the collection of the nets of the circuit. The graphs are described below:

- The graph $G_i = \langle V, E, w_i \rangle$ represents the *initial circuit*, which may not satisfy the timing constraints. The weight $w_i(e),\ \forall e \in E$ is the number of flip-flops along the wire modeled by e in G_i.
- The graph $G_p = \langle V, E, w_p \rangle$ represents the *wire-pipelined* version of the initial circuit G_i, obtained using some wire pipelining method such as [9, 10]. Although G_p satisfies the timing constraints, it may not be functionally equivalent to G_i. The weight $w_p(e),\ \forall e \in E$ is the number of flip-flops along the wire modeled by e in G_p.
- The graph $G_f = \langle V, E, w_f \rangle$ represents the *corrected wire-pipelined* circuit, obtained after altering G_p to make it functionally correct. Hence, G_f satisfies the timing constraints, and is also functionally equivalent to G_i. The weight $w_f(e)$, $\forall e \in E$ is the number of flip-flops along the wire modeled by e in G_f.

This paper accepts G_i and G_p as inputs and presents a method to obtain G_f. The input issue rate of G_i is assumed to be 1, i.e., inputs of G_i can be change every clock cycle. As was seen in section 2, any attempt to correct the functionality of G_p to obtain G_f may involve the insertion of extra flip-flops, thus increasing the area. We formulate a method to minimize the increase in area, which in detailed in section 4.2. For this purpose, we define two weight functions on E, as shown below:

- The weight $r_p(e),\ \forall e \in E$ represents the number of repeaters along the wire modeled by e in G_p.
- The weight $r_f(e),\ \forall e \in E$ represents the number of repeaters along the wire modeled by e in G_f.

We assume that all repeaters are identical and therefore have equal area. We make a similar assumption for the flip-flops as well, i.e., each flip-flop has equal area. If extra flip-flops are to be inserted along a wire, in going from G_p to G_f, some or all of the repeaters along the wires in G_p can be replaced with flip-flops. The repeaters of G_i are ignored in our model since they do not have any role in area minimization.

We extend the weight functions w_i, w_p and w_f to (simple) paths and (simple) cycles of the graphs. The weight of a path/cycle is defined as the sum of weights of all edges on the path/cycle. The weights of any edge, path and cycle in G_f must not be less than the corresponding weights in G_p, as we do not wish to *unpipeline* the wires of G_p. However, the weights w_p can be less than the corresponding weights w_i in G_i, indicating the presence of more than necessary number of flip-flops required to meet the timing requirements. Thus, for any edge or path, w_f can be less than[2] the corresponding w_i. To indicate that e is an edge from u and v in the graphs, we will use the notation $u \xrightarrow{e} v$. We will also use the terms "graph" and "circuit" interchangeably.

4. SOLUTION TECHNIQUE

4.1 Obtaining the optimal ρ

As explained in section 2, the concept of slowing down the input issue rate can be used to correct the functionality of a cycle in G_p. By specifying a restriction that inputs are not allowed to change every clock cycle, we are providing "extra" clock cycles to the cycle in G_p to complete its computations. In other words, slowdown (of input issue rate) can be thought of a compensating factor for increased cycle latencies in G_p.

Let c be any cycle of the graphs, whose latencies in G_i and G_p are $w_i(c)$ and $w_p(c)$, respectively. Consider a block on the cycle, and suppose it has an input y, not belonging to the cycle[3]. By the time the output computed by the block propagates back to itself through the other blocks of the cycle, the number of times the signal seen at y may have changed is equal to $w_i(c)$ in G_i, and $w_p(c)$ in G_p. For functional equivalence of the two circuits, the number of input changes seen at y must be identical in both circuits, equal to $w_i(c)$. This is achieved when the input y is permitted to change only every $\frac{w_p(c)}{w_i(c)}$ clock cycles in G_p. This ratio gives the slowdown $\rho(c)$ required for c in G_p. If $w_i(c)$ does not divide $w_p(c)$, then the weight $w_p(c)$ must be increased to the next higher integer. For instance, if the values of $w_i(c)$ and $w_p(c)$ are 2 and 5, respectively, then a slowdown of $\rho(c) = 3$ is required for c in G_p and the weight $w_p(c)$ must be increased to $\rho \cdot w_i(c) = 6$. The same idea can be applied to the cycle C of Figure 1, the slowdown ρ required for C is the ratio of $w_p(C)$ and $w_p(C)$, i.e., $\rho(C) = \frac{2}{1} = 2$.

In general, a circuit may have more than one cycle and each of these may require a different slowdown. The critical cycle is the cycle which requires the maximum value of slowdown. The slowdown required for this cycle is the lower bound for the slowdown required for the entire circuit G_f. If $\hat{\rho}(G_f)$, or $\hat{\rho}$ in short denotes the minimal (or optimal) slowdown required by G_f, then we have

$$\hat{\rho} = \max_{c \in \mathcal{C}} \left\{ \left\lceil \frac{w_p(c)}{w_i(c)} \right\rceil \right\}$$

where $\mathcal{C}$ is the set of cycles of the graphs.

The equation shown above represents a *maximum cycle ratio problem* (MCRP) [12] on the graphs G_i and G_p, where the time and cost of each edge $e \in E$ is given by the weights $w_p(e)$ and $w_i(e)$, respectively. One method of obtaining $\hat{\rho}$ is proposed by Lawler in [12]. The idea is to iteratively apply the Bellman-Ford algorithm [13] to find the longest paths in the graph $G_l = \langle V, E, w_l \rangle$.

$$w_l(e) \quad = \quad w_p(e) - \hat{\rho} \cdot w_i(e) \qquad \forall e \in E \tag{1}$$

If there is no cycle in G_l ($\mathcal{C} = \emptyset$), then $\hat{\rho}$ is 1, i.e., inputs can be issued every clock cycle in acyclic circuits. Otherwise, a binary search is performed to find the minimal value of $\hat{\rho}$ for which there is no positive cycle in G_l. The presence of a positive cycle in G_l indicates that for some cycle c in G_l, $\hat{\rho} \cdot w_i(c) < w_p(c)$, i.e., the slowdown required for c is greater than $\hat{\rho}$. The complexity of Lawler's method is $O(|V||E|\log(|V|w_{max}))$, where $w_{max} = \max_{e \in E} w_i(e)$. Several other more efficient ways of solving the MCRP have been proposed in the literature [14].

4.2 Obtaining a solution to G_f

4.2.1 A feasible solution

Let q and q' be any two distinct paths from the inputs of the circuits to any vertex $v \in V$. Since the inputs are issued only every $\hat{\rho}$ clock cycles in G_f, if the difference of weights of q and q' in G_i is k, then the corresponding difference in G_f must be $\hat{\rho} \cdot k$. For example, since the difference of weights of the paths $a \rightarrow y$ and $b \rightarrow z$ in ckt_i, shown in Figure 1(a) is 1, the corresponding difference must be 2 (since $\hat{\rho}$ = 2 for the circuit) in G_f for the circuit, shown in Figure 2. From this observation, we have

$$\begin{aligned} w_f(q) - w_f(q') \quad &= \quad \hat{\rho} \cdot (w_i(q) - w_i(q')) \\ \Rightarrow w_f(q) - \hat{\rho} \cdot w_i(q) \quad &= \quad w_f(q') - \hat{\rho} \cdot w_i(q') \end{aligned} \tag{2}$$

If $\mathcal{Q}_v$ is the set of all paths from the inputs to v in the graphs, then from (2), the difference of the terms w_f and $\hat{\rho} \cdot w_i$ must be equal $\forall q \in \mathcal{Q}_v$. We introduce a variable $x(v)\ \forall v \in V$ such that

$$x(v) \quad = \quad w_f(q) - \hat{\rho} \cdot w_i(q) \qquad \forall q \in \mathcal{Q}_v \tag{3}$$

[2]This not true for a cycle though. For any cycle c, $w_f(c) \geq w_i(c)$, since $\rho(c) \geq 1$.

[3]An example of such a situation is illustrated by input y in Figure 1.

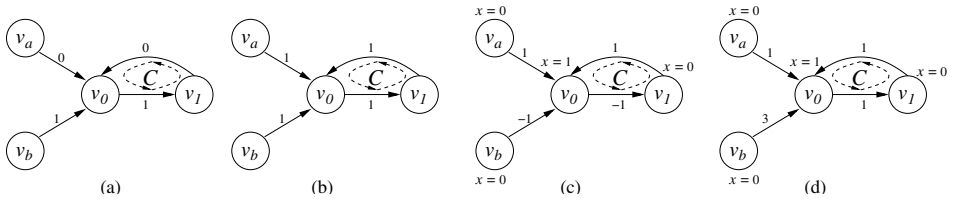

Figure 4: Illustration of the solution technique on the circuit shown in Figure 1. The numbers shown with the edges in the graphs correspond to the weights of the edges. (a) The initial circuit (G_i) depicting ckt_i. (b) The wire pipelined circuit (G_p), depicting ckt_p. (c) The corresponding graph G_l. The optimal slowdown, $\hat{\rho}$ is 2. The number shown above each vertex in G_l is the x value for that vertex. (d) A Solution (G_f). The weights w_f shown with the edges are obtained by using (7).

We also have $w_f(q) \geq w_p(q)$ for all $q \in \mathcal{Q}_v$. From this and (3), the following can be deduced:

$$x(v) \quad \geq \quad w_p(q) - \hat{\rho} \cdot w_i(q) \qquad \forall q \in \mathcal{Q}_v \tag{4}$$

Let q_u be any path starting from the inputs, ending at vertex u. For $u \xrightarrow{e} v$, we can form a path q_v ending at v by adding e to q_u. Therefore, we have

$$\begin{aligned} w_p(q_v) &= w_p(q_u) + w_p(e) \\ w_i(q_v) &= w_i(q_u) + w_i(e) \\ \text{and} \quad w_f(q_v) &= w_f(q_u) + w_f(e) \end{aligned} \tag{5}$$

From (4) and (5), we have

$$\begin{aligned} (x(v) - x(u)) \geq (w_p(q_v) - w_p(q_u)) + \hat{\rho} \cdot (w_i(q_v) - w_i(q_u)) \\ \Rightarrow x(v) \geq x(u) + (w_p(e) - \hat{\rho} \cdot w_i(e)) \end{aligned} \tag{6}$$

From (6), it is evident that $x(v)$ is the weight of the longest path to v in G_l, defined in section 4.1. When there are no positive cycles in G_l, longest paths are well defined and the Bellman-Ford algorithm outputs the x values of the vertices. Therefore, solving the MCRP by Lawler's method also finds the x values, along with $\hat{\rho}$. We will now show that the weights w_f of G_f can be determined from the x values and $\hat{\rho}$ obtained by solving the MCRP on G_i and G_p. From (3) and (5), we have

$$\begin{aligned} w_f(q_v) &= x(v) + \hat{\rho} \cdot w_i(q_v) \\ \Rightarrow w_f(q_u) + w_f(e) &= x(v) + \hat{\rho} \cdot (w_i(q_u) + w_i(e)) \\ \Rightarrow w_f(e) &= (x(v) - x(u)) + \hat{\rho} \cdot w_i(e) \end{aligned} \tag{7}$$

In (7), the weights w_f are expressed in terms of x values and $\hat{\rho}$. To summarize, the following steps are involved in obtaining G_f.

1. Solve the MCRP to obtain $\hat{\rho}$ and the x values.
2. From the $\hat{\rho}$ and the x values computed in step 1, determine the weights w_f of G_f using (7).

LEMMA 1. *Let $(G_f = \langle V, E, w_f \rangle, \rho \geq \hat{\rho})$ be a solution to $\langle G_i, G_p \rangle$. Then for any cycle c in the circuit, we have*

$$w_f(c) \quad = \quad \rho \cdot w_i(c)$$

The proof of Lemma 1 is omitted due to space limitations. The lemma indicates that all cycle latencies are increased by a factor of ρ in G_f. This shows that G_f represents a pipelined version of G_i, retaining its functionality if the inputs are issued only every ρ clock cycles. It produces outputs every ρ clock cycles.

We demonstrate the solution technique on the circuit shown in Figure 1. Figures 4(a) and (b) show the graph models G_i and G_p, for the circuits ckt_i and ckt_p, shown in Figures 1(a) and (b), respectively. The blocks B_0 and B_1, and the inputs a and b are modeled as the vertices v_0, v_1, v_a, v_b, respectively. The graphs have one cycle $C = v_0 \rightarrow v_1 \rightarrow v_0$. We have seen at the beginning of this section that the optimal slowdown required for the circuit is 2, i.e., $\hat{\rho} = 2$. Figure 4(c) shows the graph G_l obtained by computing the edge weights using (1). For $\hat{\rho} = 2$, it can be observed that the weight of C in G_l is 0, which indicates that the longest paths are well defined in G_l. The x values of the vertices are shown in Figure 4(c). The solution obtained by using the x values from Figure 4(c) is shown in Figure 4(d). It can be seen that the graph G_f of Figure 4(d) is identical to the circuit ckt_f of Figure 2.

4.2.2 A minimum area solution

The solution technique presented in the previous section only finds a feasible solution, and does not consider minimization of the area increase, incurred due to the possible insertion of extra flip-flops. One way of minimizing the number of extra flip-flops is to retime some or all of the extra flip-flops out of the wires of the circuit, as illustrated in Figure 5. In this section, we will extend the solution technique to incorporate area minimization and formulate the problem as an Integer Linear Program (ILP) and then describe a method to solve the ILP efficiently.

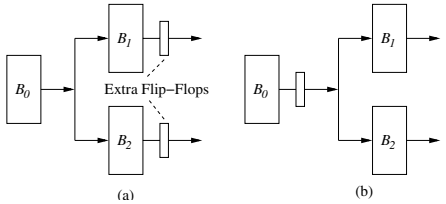

Figure 5: Illustration of area minimization on a portion of a circuit. (a) A solution to the problem requires one extra flip-flop each on the outgoing edges of B_1 and B_2, respectively. (b) The two flip-flops are moved over the blocks B_1 and B_2 to the outgoing edge of B_0, which reduces the flip-flop count by one.

Formulation as an ILP

In section 4.2.1, the x values are computed as the longest path weights in G_l. However, the slacks in the longest path constraints (henceforth referred to as latency constraints) (6) allow a range of permissible values for x. This flexibility enables the movement of flip-flops across vertices, which is exploited for area minimization.

We define the area of the edge e in G_f, $a_f(e)$, as the area of the repeaters and flip-flops along e. If $area$ is the total area of the repeaters and flip-flops of G_f, and w_a and r_a are the areas of a single flip-flop and repeater, respectively, then for any $\rho \geq \hat{\rho}$,

$$\begin{aligned} a_f(e) &= w_f(e) \cdot w_a + r_f(e) \cdot r_a \quad \forall e \in E \\ \text{and} \quad area &= \sum_{e \in E} a_f(e) \end{aligned} \tag{8}$$

In the event of adding extra flip-flops to the edge e, some or all of the repeaters present along e in G_p can be replaced with flip-flops. In this paper, we assume that each extra flip-flop can replace one repeater from the edge. The available number of slots, i.e., repeaters along the edge e in G_p is given by $r_p(e)$ and the num-

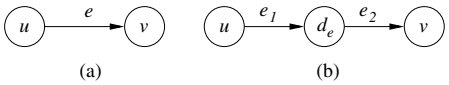

Figure 6: Insertion of a dummy node d_e on an edge $e \in E$.

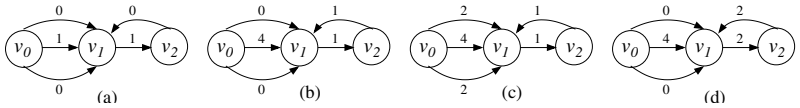

Figure 7: Optimal $\hat{\rho}$ may not mean minimum area. It is assumed that the circuits do not have repeaters. The number shown with each edge in the graphs denotes the flip-flop count of the edge. (a) Initial circuit. (b) Wire-pipelined circuit. (c) A minimum area solution for $\rho = \hat{\rho} = 2$: number of flip-flops = 10. (d) A minimum area solution for $\rho = 4$: number of flip-flops = 8.

ber of extra flip-flops to be added to the edge e in G_f is given by $extra(e) = w_f(e) - w_p(e)$. If $extra(e)$ exceeds $r_p(e)$, then all of the $r_p(e)$ will be removed and replaced with flip-flops. In such a scenario, the repeater count, $r_f(e)$ will be 0. Otherwise, $r_f(e)$ will be equal to the remaining number of repeaters of G_p, after some of them were replaced by extra flip-flops. Therefore, $r_f(e)$ can be expressed as follows:

$$r_f(e) \quad = \quad max\{r_p(e) - (w_f(e) - w_p(e)), 0\} \qquad (9)$$

The objective of the minimum area solution is to minimize $area$ given by (8) subject to the constraints (6) and (9), which can be formulated as an ILP, by expressing (9) as two linear constraints.

Solving the ILP

Solving an ILP is generally *NP–complete*, unless the problem exhibits integral polytope structure. The ILP described in the previous section can be formulated as an instance of the dual of the Minimum Cost Network Flow (MCF) problem [15], which exhibits integral polytope structure and can be efficiently solved. This can be accomplished by eliminating the weights r_f from the ILP formulation. For each edge $e \in E$, where $u \xrightarrow{e} v$, we add a dummy vertex d_e and split e into two edges, e_1 and e_2, such that $u \xrightarrow{e_1} d_e$ and $d_e \xrightarrow{e_2} v$, as shown in Figure 6. The edge e_1 models the case where the extra flip-flops to be inserted on e replace the repeaters of e. Inserting a flip-flop on e_1 increases the area of e by $w_a - r_a$. The edge e_2 models the case where more than $r_p(e)$ extra flip-flops are to be inserted on e. The first $r_p(e)$ extra flip-flops to be inserted on e are assigned to e_1 and the remaining to e_2. Therefore, $w_f(e_2)$ will be strictly positive only when the number of extra flip-flops exceeds $r_p(e)$. Inserting an extra flip-flop on e_2 increases the area of e by w_a. We have,

$$\begin{aligned} w_f(e) &= w_f(e_1) + w_f(e_2) \\ w_f(e_1) &\leq r_p(e) + w_p(e) \qquad (10) \\ r_f(e) &= r_p(e) - (w_f(e_1) - w_p(e)) \qquad (11) \end{aligned}$$

The weights r_f can now be eliminated from the ILP using (11). The following latency constraints on e_1 and e_2 can be inferred from the above equations.

$$\begin{aligned} x(d_e) &\geq x(u) + (w_p(e) - \rho \cdot w_i(e)) && (w_f(e_1) \geq w_p(e)) \\ x(v) &\geq x(d_e) && (w_f(e_2) \geq 0) \\ x(d_e) &\leq x(u) + (r_p(e) + w_p(e) - \rho \cdot w_i(e)) && (\text{from } (10)) \end{aligned}$$

It can be observed that the first two inequalities above add up to obtain the constraint (6) on e. We now find the expression for $area$.

$$\begin{aligned} a_f(e) &= w_f(e) \cdot w_a + (r_p(e) + w_p(e) - w_f(e_1)) \cdot r_a \\ &= (x(v) - x(u)) \cdot w_a - (x(d_e) - x(u)) \cdot r_a + \rho \cdot const. \\ &= x(v) \cdot w_a - x(u) \cdot (w_a - r_a) - x(d_e) \cdot r_a + \rho \cdot const. \end{aligned}$$

$$area = \sum_{v \in V \bigcup V_d} (k_v \cdot x(v)) + \rho \cdot const.$$

where V_d is the set of dummy vertices, and if $FO(v)$ and $FI(v)$ are the number of outputs and inputs of $v \in V$, respectively,

$$k_v = \begin{cases} FI(v) \cdot w_a - FO(v) \cdot (w_a - r_a) & : \quad v \in V \\ -r_a & : \quad v \in V_d \end{cases}$$

A minimum area solution to G_f is formulated as the ILP

$$\begin{aligned} \text{Minimize} \quad area &= \sum_{v \in V \bigcup V_d} (k_v \cdot x(v)) + \rho \cdot const \\ \forall e \in E \qquad & s.t. \quad u \xrightarrow{e} v \\ x(u) - x(d_e) &\leq \rho \cdot w_i(e) - w_p(e) \\ x(d_e) - x(v) &\leq 0 \\ x(d_e) - x(u) &\leq r_p(e) + w_p(e) - \rho \cdot w_i(e) \end{aligned}$$

For a constant ρ, the preceding ILP is an instance of the dual of the minimum cost flow problem, which can be efficiently solved by several methods such as the network simplex method [15]. As before, the weights w_f can be computed using (7). There is a minimum area solution for each value of $\rho \geq \hat{\rho}$. In addition, the minimum area solution for $\hat{\rho}$ may not be a global minimum solution, as demonstrated in Figure 7. However, in most cases, maximizing throughput (or minimizing ρ) is the primary objective, rather than minimizing area. In such a scenario, the ILP is solved for $\rho = \hat{\rho}$, which is obtained by solving the MCRP, as detailed in section 4.1.

In general, it is not easy to determine how many repeaters can be removed from a wire without worsening the clock period, when an extra flip-flop is inserted. In the above procedure, it was assumed that every extra flip-flop replaces one repeater on the wire. This can easily be extended to other complex flip-flop repeater models. One such a model can be as follows. For a wire, the number of repeaters required for a range of number of flip-flops (inserted on the wire) can be specified. Beyond a certain number of flip-flops, no repeaters may be required to meet the timing requirements.

5. EXPERIMENTAL RESULTS

For experimentation, we have used the ISCAS benchmark suite [16]. An operating frequency of 3GHz was chosen for the system and the target technology chosen has a feature size of 70nm. After finding a placement using Capo [17], the area of the circuits was scaled to 4.30cm^2 to mimic the layout of a realistic chip. For smaller layouts, the wire lengths are not long enough to be pipelined. The dimensions of the circuits were scaled accordingly. Each gate in the original circuit is assumed to be a combinational functional block, and each wire is assumed to be latched immediately, after it leaves the block. In addition, none of the global wires is assumed to have flip-flops. For the wire delays, the projections for a 2cm global wire made in [2] were used, where the delay of an optimized 2cm wire in 70nm technology is projected to be 0.67ns. The delays of the wires of the test circuits were determined by assuming a linear relationship between the delay of a wire and its length, which is reasonable for buffered interconnects. It is also

Circuit	$\|V\|$	$\|E\|$	G_p		G_f		$\hat{\rho}$	Incr	Time
			Rptrs	Flops	Rptrs	Flops		(%)	(sec)
s27	15	18	21	19	18	22	1	5.1	0.1
s344	110	210	261	229	193	327	2	17.8	0.1
s349	114	215	238	231	212	260	1	4.6	0.1
s1196	360	836	1857	1108	1576	1459	1	10.3	1
s1238	389	925	2076	1228	1672	1794	1	16.1	1
s1423	449	913	1112	998	750	1504	2	20.5	1
s1494	364	1104	2991	1571	2502	2176	2	11.8	1
s13207	2014	3759	4825	4118	3327	5992	2	17.2	1
s15850	3504	7215	8892	7774	6010	11325	2	17.3	2
s38417	8029	17646	27572	20411	21297	28996	2	15.8	14
s38584	9616	22515	35831	26170	27240	36835	2	14.4	24

Table 1: Experimental results for ISCAS benchmarks.

Circuit	MaxLen (cm)	Delay (ns)	MaxFreq (GHz)	S_{G_p}	$\hat{\rho}$	S_{G_f}
s27	1.22	0.41	2.46	1.22	1	1.22
s344	1.75	0.59	1.71	1.76	2	0.88
s349	1.45	0.49	2.06	1.46	1	1.46
s1196	3.23	0.92	1.08	2.78	1	2.78
s1238	2.91	0.98	1.02	2.93	1	2.93
s1423	2.66	0.89	1.12	2.67	2	1.34
s1494	2.83	0.95	1.05	2.85	2	1.43
s13207	3.18	1.06	0.94	3.19	2	1.60
s15850	2.71	0.91	1.14	2.73	2	1.37
s38417	3.64	1.22	0.82	3.65	2	1.83
s38584	3.88	1.30	0.77	3.90	2	1.95

Table 2: Performance issues with wire pipelining.

assumed that a 2cm wire has 10 repeaters, and accordingly the repeater counts of the wires of the circuit were determined. The area of a flip-flop was assumed to be twice that of a repeater.

First, the optimal slowdown, $\hat{\rho}$ was obtained for each circuit by solving the MCRP, as explained in section 4.1. Later, the ILP was solved using the network simplex implementation of [18] to obtain a minimum area solution subject to the $\hat{\rho}$ for each circuit. The experiments were performed on a 2.4GHz Pentium 4 machine with 1GB RAM. The results obtained for different benchmarks are shown in Table 1. The labels **Rptrs** and **Flops** denote the number of repeaters and flip-flops, respectively, listed for both circuits G_p and G_f. It can be seen that the number of repeaters decreases in G_f, since some of the repeaters in G_p are replaced by flip-flops in G_f. For circuits such as s1238 and s1196, a slowdown of 1 indicates that none of the wires forming cycles in those circuits were long enough to be pipelined. The last column lists the percentage increase in the area of the repeaters and flip-flops in G_f. The area is calculated as the sum of the areas of the flip-flops and repeaters, which were normalized to 2 and 1, respectively. The run times are in the order of a few seconds, as shown in the table.

Table 2 captures the speedup obtained by wire pipelining. The entries of columns 2–4 are related to the circuit G_i for each benchmark. The column labeled **MaxLen** shows the maximum wire length of the global interconnects for each benchmark, and corresponding wire delay is shown in column 3. Column 4 lists the upper bound on the operating frequency of G_i for each benchmark, which is computed as the reciprocal of the delay shown in column 3. The column labeled $\mathbf{S_{G_p}}$ shows the frequency speedup achieved by performing wire pipelining on G_i for a clock frequency of 3GHz. However, the frequency speedup of the wire-pipelined circuit, G_p (which may be functionally incorrect) may not entirely translate into the throughput speedup obtained for the corrected wire-pipelined circuit, G_f, since the possibility of increased cycle latencies in G_p will enforce a slowdown of $\hat{\rho}$ in the input issue rate in G_f. The column $\mathbf{S_{G_f}}$ shows the actual throughput speedup achieved by G_f, where $\mathbf{S_{G_f} = S_{G_p}/\hat{\rho}}$.

It can be observed from Table 2 that the throughput speedup achieved is less than one for the circuit s344 which indicates that wire pipelining has resulted in throughput degradation for this circuit. The slowdown required can be improved by using better objective functions in placement.

Although wire pipelining causes a degradation in performance for some circuits, there could be several system-wide reasons for having a higher clock frequency. Typically, decision on the operating frequency is made at the system level and is handed down to the designer to implement, who tries to ensure best possible performance under this decision.

6. CONCLUSION

This paper has presented an approach to solve the problems created by wire pipelining. The method presented in this paper also finds the optimal value of input issue rate slowdown required for the circuits, which directly affects the throughput. The problem is formulated as an instance of the dual of minimum cost flow problem, to incorporate the minimization of area increase, incurred due to the insertion of extra flip-flops. Though wire pipelining improves overall throughput of most circuits, it may degrade the throughput for some circuits. However, this is still a useful solution since clock frequencies are typically decided by system-wide considerations, and the task of the designer is to obtain the best achievable performance under such system-level constraints.

7. REFERENCES

[1] S. Borkar, "Obeying Moore's law beyond 0.18 micron," in *Proceedings of the International ASIC/SOC Conference*, pp. 26–31, Sep. 2000.

[2] J. Cong, "An interconnect-centric design flow for nanometer technologies," in *Proceedings of the IEEE*, vol. 89, pp. 505–528, April 2001.

[3] D. M. Chapiro, *Globally-Asynchronous Locally-Synchronous systems*. PhD thesis, Stanford University, Stanford, California, Oct. 1984.

[4] J. N. Seizovic, "Pipeline synchronization," in *Proceedings of the IEEE ISASYNC*, pp. 87–96, Nov. 1994.

[5] D. S. Bormann and P. Y. K. Cheung, "Asynchronous wrapper for heterogeneous systems," in *Proceedings of the IEEE ICCD*, pp. 307–314, Oct. 1997.

[6] L. P. Carloni and A. L. Sangiovanni-Vincentelli, "Performance analysis and optimization of latency insensitive systems," in *Proceedings of the ACM DAC*, pp. 361–367, Jun. 2000.

[7] C. E. Leiserson *et al.*, "Optimizing synchronous circuitry by retiming," in *Proceedings of the Third Caltech Conference on VLSI*, pp. 87–116, Mar. 1983.

[8] C. Lin and H. Zhou, "Retiming for wire pipelining in system-on-chip," in *Proceedings of the IEEE ICCAD*, pp. 215–220, Nov. 2003.

[9] P. Cocchini, "Concurrent flip-flop and repeater insertion for high performance integrated circuits," in *Proceedings of the IEEE ICCAD*, pp. 268–273, Nov. 2002.

[10] S. Hassoun *et al.*, "Optimal buffered routing path constructions for single and multiple clock domain systems," in *Proceedings of the IEEE ICCAD*, pp. 247–253, Nov. 2002.

[11] N. Sherwani, *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Boston, Massachussets, 2nd ed., 1995.

[12] E. L. Lawler, "Optimal cycles in doubly weighted directed linear graphs," in *Proceedings of the International Symposium on Theory of Graphs*, pp. 209–213, Jul. 1966.

[13] T. H. Cormen *et al.*, *Introduction to Algorithms*. MIT Press, Cambridge, Massachussets, 1st ed., 1989.

[14] A. Dasdan *et al.*, "Efficient algorithms for optimum cycle mean and optimum cycle cost to time ratio problems," in *Proceedings of the ACM DAC*, pp. 37–42, Jun. 1999.

[15] M. S. Bazaara *et al.*, *Linear Programming and Network Flows*. John Wiley and Sons, New York, 2nd ed., 1990.

[16] CBL: NCSU, "ISCAS89 Benchmark Suite." Available at `http://www.cbl.ncsu.edu/CBL_Docs/iscas89.html`.

[17] A. Caldwell *et al.*, "Capo: A large-scale fixed-die placer." Available at `http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/Capo/`.

[18] A. Loebel, "MCF Version 1.2 - A network simplex implementation ." Available at `http://www.zib.be`.