Retiming of Two-Phase Latch-Based Resilient Circuits*†

Hsiao-Lun Wang[1] hsiaoluw@usc.edu
Minghe Zhang[2] zmh13@mails.tsinghua.edu.cn
Peter A. Beerel[1] pabeerel@usc.edu

[1]Ming Hsieh Dept. of Electrical Engineering, University of Southern California, Los Angeles, CA
[2]Department of Microelectronics and Nanoelectronics, Tsinghua University, Tsinghua, China

ABSTRACT

Timing resilient design has shown significant promise in mitigating the excess margins associated with rare worst-case data and increased process, voltage, and temperature (PVT) variations. However, resilient circuits need error detecting sequential logic (EDL) to detect timing errors which represents area and power overhead. This article proposes a new network-simplex-based retiming method for two-phase latch-based resilient circuits to reduce the overhead of the combination of normal and error detecting latches. Our experimental results show that our method is computationally efficient and reduces the cost of the sequential elements by an average of up to 20% when compared to traditional methods.

1. INTRODUCTION

Traditional synchronous designs must incorporate timing margin to ensure correct operation under worst-case delays caused by process, voltage, and temperature (PVT) variations and cannot take advantage of average-case path activity [3]. This is particularly problematic in low-power low-voltage designs, as performance uncertainty due to PVT variations grows from as much as 50% at nominal supply to around 2,000% in the near-threshold domain [6]. To address this problem, many design techniques for timing resilient circuits have been proposed. For example, canary FFs attached to replica critical paths predict when the design is close to a timing failure [22]. Designs can then adjust their supply voltage or clock frequency either statically or dynamically to ensure correct operation at the edge of failure [17]. Other resilient design techniques use extra logic to detect and recover from timing violations [4, 5, 7, 8]. These techniques use a variety of error-detecting latches and/or flip-flops, initiate replay-and-recovery or slow-down/stall the pipeline in the case of errors, and span both synchronous and asynchronous design styles. All resilient designs exhibit higher performance when there are no timing errors and gracefully slow down in the presence of timing errors, thus achieving higher average-case performance than traditional worst-case designs. This additional performance can often be traded off for lower power via further voltage scaling.

Some EDA techniques have been proposed to minimize the probability of timing errors [10, 13, 23]. Of particular importance to this paper, however, is that the error detecting logic (EDL) that is necessary to enable resilient designs represents area and power overhead when compared with traditional worst-case designs. Thus the EDL must represent a relatively small fraction of the design to not overshadow the obtained average-case performance benefits. Circuit and EDA techniques to minimize the EDL overhead will thus be important for these techniques to flourish. Some proposed a resynthesis technique to reduce overheads [8–10] in which near-critical-paths are sped-up by re-running logic synthesis with a tighter max delay constraint to reduce the EDL needed at the cost of increased logic area.

The closest to our work is that proposed by [14] in which they explored retiming resilient designs, i.e., relocating sequential gates to reduce the amount of EDL without changing the combinational logic. The authors in that work assumed flop-based resilient designs and minimize the number of suspicious flops that need to be error-detecting. Our work is focused on latch-based resilient circuits, for which both synchronous [7, 11] and asynchronous [8] styles have been proposed. In particular, we assume a flow in which flops are converted into master-slave latches and the slave latches are retimed. Latch-based resilient circuits have higher hold margins but introduce additional challenges in managing error-detecting overhead associated with the doubling of the sequential elements. In particular, as proposed in [8, 11], this work assumes that only a subset of the latches are error-detecting and that the remaining latches time-borrow to reduce overhead while maintaining timing resiliency. In particular, we only allow the retimed slave latches to time borrow and do not retime the master latch to avoid challenges caused by changing the initial state of the circuit [21].

We propose to extend the traditional retiming algorithm used in [7, 8, 11] by explicitly modeling the cost of the error-detecting latches. Our model captures the fact that different retiming solutions lead to different subsets of master latches that are near-critical and thus that need to be error-detecting. We first present an Integer Linear Programming (ILP) formulation that solves the modified retiming problem minimizing sequential area and then transform this formulation into a network flow problem for which exact efficient solutions exist. Our experimental results on a wide variety of benchmark circuits show that our method can reduce an average of up to 20% of the cost of the sequential elements, with all examples completing within 15 minutes of CPU time.

The remainder of the paper is organized as follows. Section 2 introduces the background of retiming and resilient circuits. Section 3 presents the problem definition and our approach. Then, Section 4 develops our network simplex approach. Section 5 shows our experimental results, followed by a brief summary in Section 6.

2. BACKGROUND

2.1 Two-Phase Latch-Based Resilient Circuits

The clock model [18] of latch-based circuit design often assumes k-phases with k periodic clock signals, $\Pi = \langle \phi_1, \gamma_1, \phi_2, \gamma_2, \ldots, \phi_k, \gamma_k \rangle$, where ϕ_i the transparent window of phase i, and γ_i is duration from the falling edge of phase i to the rising edge of

*This work was supported in part by NSF Grant #1619415.

†Peter A. Beerel is also Chief Scientist at Reduced Energy Microsystems.

DAC '17, June 18-22, 2017, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/3061639.3062312

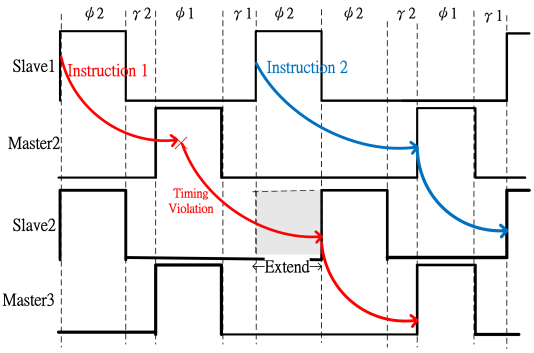


Figure 1: Timing of a two-phase latch-based resilient circuit.

phase i+1. For a two-phase latch-based design, $\Pi = \langle \phi_1, \gamma_1, \phi_2, \gamma_2 \rangle$. As in [8, 11], we assume a symmetric clocking scheme for resilient latch-based design in which $\phi_1 = \phi_2$ and $\gamma_1 = \gamma_2$. The resiliency window [8, 11] is ϕ_1. Fig. 1 shows the clocking diagram of a two-phase latch-based resilient circuit. Assuming ideal clock trees, the maximum delay P of data path between master stages in the circuit is $\phi_1 + \gamma_1 + \phi_2 + \gamma_2 + \phi_1 = \Pi + \phi_1$. However, the period Π of such circuits is only $\phi_1 + \gamma_1 + \phi_2 + \gamma_2$. If there are data transitions during the resiliency window of a master latch, the rising edge of the next pipeline stage's clock must be delayed to ensure the setup time requirement are satisfied for this and subsequent stages. This imposes timing constraints on the error detection logic that forces designs to smartly group error-detecting latches into manageable clusters [8] and, in the case of synchronous designs, impose a complex local clock stalling scheme [7]. Note also that time borrowing is allowed for slave stages and only master stages can be error detecting. This reduces the overhead of EDL while simultaneously reducing the complexity of the timing constraints [8, 11].

2.2 Error Detecting Latches

Timing resilient circuits rely on error detecting latches/flops to detect whether the data remains constant during the resilient window. Fig. 2 shows two types of error detecting latches (EDL) [3]: a) a time-borrowing latch with a shadow master-slave Flip-Flop (MSFF) in which the MSFF samples the value of data at the rising edge of resilient window and an XOR gate compares the value of sampled value and the data during the resiliency window and b) a transition detecting time-borrowing latch (TDTB) which consists of a conventional latch, an XOR gate to detect transitions, and an asymmetric C-element to hold the value of the error. The error signals of each error detecting latches within a pipeline stage are collected with a OR gate tree to produce an error signal for the entire stage. Different EDLs have different amounts of area overhead compared to a normal latch, which in this paper is denoted c. For the purposes of this paper we consider a range of c from 0.5 to 2, similar to [9], representing the fact that the amortized area of different proposed EDLs can range from 50% to 2X larger than a normal latch.

2.3 Min-Area Retiming

There are two traditional objectives for retiming algorithms, minimizing the clock period [12, 19] and minimizing the number of sequential elements [12, 15, 16]. We now review the traditional min-area re-timing algorithm because it is more closely related to our work. As in [12], a sequential circuit can be described as $G(v, e)$ where each v represents a combinational gate and each directed edge e_{uv} represents a connection between gate u and gate v. The weight of the edge $w(e_{uv})$ represents the number of FFs or latches between gate nodes. Each vertex has a fixed delay $d(v)$. A special vertex, the host vertex, is introduced into the graph, with edges from the host vertex to all primary inputs of the circuit, and edges from all primary outputs to the host vertex.

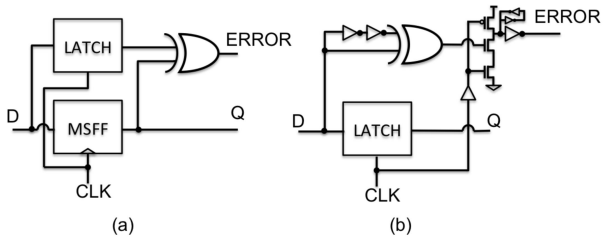


Figure 2: Two kinds of error detecting latches [3].

Two additional variables D_{uv} and W_{uv} are defined as follows.

$$W_{uv} = \min_{\forall p, u \rightsquigarrow v} w(p) \tag{1}$$

$$D_{uv} = \max_{\forall p, u \rightsquigarrow v,\ w(p) = W_{uv}} d(p) \tag{2}$$

In the equations above, p represents a path from gate u to gate v, $w(p)$ represents the total number of latches along p and $d(p)$ is the delay from u to v via p. $r(u)$ represents the number of FFs or latches that are retimed from the output of gate u towards the inputs of gate u. $w_r(e_{uv}) = w_{uv} - r(u) + r(v)$ denotes the number of FFs or latches on edge $e(u, v)$ after retiming. The objective function and constraints of the classic retiming algorithm [12] for flip-flop based design are as follows.

$$\begin{aligned} \min \sum_{v \in V} \left[\sum_{\forall u \in FI(v)} \beta(e_{uv}) - \sum_{\forall u \in FO(v)} \beta(e_{uv}) \right] r(v) \\ s.t. \quad r(u) - r(v) \le w(e_{uv}) \qquad \forall e_{uv} \in E \\ r(u) - r(v) \le W(u, v) - 1 \quad \forall D(u, v) > P \end{aligned} \tag{3}$$

Here, P is the cycle time of clock and the path delay between any two flops should be less than P to avoid violating setup timing constraints. In addition, for a gate u with fanout k, $\beta(e_{uv}) = \frac{1}{k}$, is the cost coefficient of a flip-flop on edge e_{uv} that models fanout sharing with the introduction of pseudo nodes [12].

To simplify the constraints, [15] re-writes them as follows.

$$\begin{aligned} \min \sum_{v \in V'} \left[\sum_{\forall u \in FI(v)} \beta(e_{uv}) - \sum_{\forall u \in FO(v)} \beta(e_{vu}) \right] r(v) \\ s.t. \qquad r(u) - r(v) \le c_{uv} \quad \forall (u, v) \in C' \\ L_u \le r(u) \le U_u \quad \forall u \in V' \end{aligned} \tag{4}$$

Here, L_u and U_u are the lower and upper bounds of the number of latches or flip-flops that can be moved backward through gate u without violating the timing constraints [15] and their values are calculated through ASAP and ALAP skews [20]. C' is the set $\{(i, j) \in C | U_i - L_j > c_{ij}\}$, where C is the constraints set of (3), and $V' = \{v \in V | U_v \neq L_v\}$.

3. THE PROBLEM AND OUR APPROACH

3.1 Problem Statement

As mentioned earlier, in our latch-based retiming algorithm we assume the location of the master latches are fixed and move only the slave latches as illustrated in Fig. 3. The original position of slave latches before retiming are at the output of their associated master latches. The combinational logic path delay between the master latches in stage 1 (M1) and the master latches in stage 2 (M2) is unaffected by the retiming because the positions of master latches in M1, M2 are fixed. However, the different positions of the slave latches change the arrival time of the data at the M2 master latches. This is because the slave latches become transparent at time $\phi_1 + \gamma_1$ and only then start propagating signals through the subsequent combinational logic as illustrated in

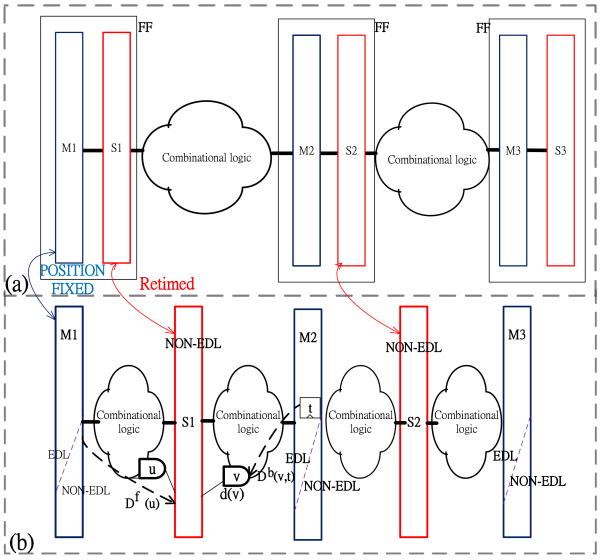


Figure 3: (a) Original FF based design. (b) Two-phase latch-based resilient circuit obtained after retiming. Our algorithm minimizes the overall cost of the sequential elements by choosing the optimal retimed position for all slave latches.

Fig 1. The delay from source master latches s through a repositioned slave latch on edge $e(u, v)$ to a given destination master t can be expressed as

$$A(u, v, t) = \max\{\phi_1 + \gamma_1, D^f(u)\} + d(v) + D^b(v, t) + D_l \quad (5)$$

Here, $d(v)$ is the delay of gate v and $D^f(u)$ is the maximum delay from any master latch in the current stage to the slave latch at output of gate u, obtained through classic static timing analysis (STA). Similarly, $D^b(v, t)$ is the maximum delay from a slave latch at output of gate v to the destination master latch t. It is the delay calculated backwardly starting from master t to slave v. D_l is the D to Q delay of a latch. Figure 3(b) shows the delay from M1 to a certain master latch t in M2 passing through a slave latch between gate u, gate v. The arrival time at the master latch t is maximum of (5) over those slave latches in its fan-in cone.

For a retiming to be valid, there must be exactly one slave latch along any path from a master s to a master t. In Fig. 3, when we want to find the optimal positions of latches in S1, the source master s is in M1, and destination master t is in M2. Similarly, when we want to find the optimal positions of latches in S2, the master s is in M2 and master t is in M3. As Fig. 1 shows, if the data changes during the resilient window ϕ_1, the clock signal of the next stages is delayed by ϕ_1. The data of a master latch s has always at least time $\phi_1+\gamma_1+\phi_2$ to propagate to the slave latches of the next stage even if the data is still changing during the resilient window. For this reason, we can view that a master s always propagates data at time 0, and that slave latches are transparent during time $\phi_1 + \gamma_1$ to $\phi_1 + \gamma_1 + \phi_2$. In Fig. 3, we can assume M2 always propagate data at time 0. The combinational logic and sequential elements before stage M2 do not affect the arrival time at stage M3. As a result, whether the master latches in M3 are EDL or not depends on the positions of the slave latches S2 and the combinational logic between M2 and M3. It does not depend on the positions of the slave latches of S1 or combinational logic between M1 and M2. Similarly, whether master latches in M2 are EDL only depends on the positions of S1 and combinational logic between M1 and M2. In other words, each pipeline stage can be retimed independently without any loss of optimality.

For a master latch t, the fan-in cone of t is denoted as $FIC(t)$. If $\exists v \in FIC(t)$, $D^f(v) + D^b(v, t) > \Pi$, then master latch t must be error detecting no matter where the slave latches are re-timed. For other master latches, however, the decision as to whether they must be error-detecting or not depends on the location of the slave latches. In particular, if the slave latches v are close to the master latch s and $D^b(v, t)$ is large, then a master latch t may have an arrival time that exceeds Π forcing it to be error-detecting. If, however, we move the slave latches forward through more of the combinational logic, the total delay of signals between two master stages s and t may decrease due to lower $D^b(v, t)$ to less than or equal to Π, and the error detecting latch can be set to non-error detecting. For this reason, the position of slave latches is important to reduce the total number of error-detecting latches. We refer to master latches whose error-dependent status is retiming dependent as *target* master latches.

Note, there is also a maximum amount we can move any slave latch through the combinational logic that is dictated by the constraint that the data needs to pass through the slave latch before it becomes opaque. This is sometimes referred to as the time borrowing constraint. Assuming the slave latch has a transparency phase of time ϕ_2, if a slave latch is placed at gate v, we have

$$D^f(v) \leq \phi_1 + \gamma_1 + \phi_2. \quad (6)$$

Similarly, the data from slave stages v should arrive at termination master latches t before the time master latches t goes opaque. That is, if a slave latch is placed at node v, we have following constraint for all terminating master latches t,

$$D^b(v, t) \leq \phi_2 + \gamma_2 + \phi_1 \quad (7)$$

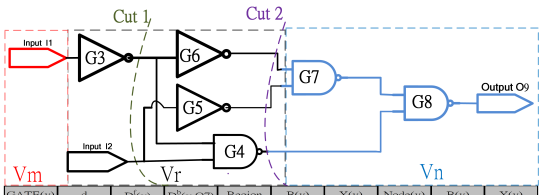
The goal of our work is to re-time the slave latches to minimize the cost of the sequential logic, including the total number of slave and master latches and the error-detecting overhead, while still satisfying the above constraints.

Fig. 4 is an example used to illustrate our problem. The circuit is cut at the flip-flops turned master-slave latches such that the primary inputs (PI) of the circuit include outputs of the (fixed) master latches and the primary output (PO) of the circuit can in reality be the input of a (fixed) master latch as the combinational logic cloud in Fig. 3(a). Note that slave latches before retiming are at the inputs of the circuit. We find a new cut to place the slave latches in the combinational logic cloud.

Assume $\phi_1 = \gamma_1 = \phi_2 = \gamma_2 = 2.5$ and $D_l = 0$. The delay of each gate is shown in column d, and $D^f(v), D^b(v, t)$ of each gate is shown in the third and fourth column respectively of lower left-hand table in Fig. 4. The slave latches cannot be placed at output of $G7$ or $G8$ since the forward delay $D^f(G7), D^f(G8)$ larger than $\phi_1 + \gamma_1 + \phi_2 = 7.5$ which violates Constraint (6). Slave latches can neither be placed at $I1$ since the backward delay from master $O9$, $D^b(I1, O9)$ is larger than $\phi_2 + \gamma_2 + \phi_1 = 7.5$ which violates Constraint (7). If we choose Cut1 to place the slave latch, the arrival time at $O9$ is $max(D^f(G3), \phi_1 + \gamma_1) + D^b(G6, O9) + d(G6) = 12$. Since this is larger than $\Pi = 10$, $O9$ should be an error detecting master latch. If we choose Cut2 to place the slave latches, there are two paths from inputs to output $O9$, the longest delay goes through $G6$, and arrival time at $O9$ is $\max(D^f(G6), \phi_1 + \gamma_1) + D^b(G7, O9) + d(G7) = 9$ which is less than $\Pi = 10$. We can analyze the total sequential elements cost and compare them between the two cuts. Cut1 needs two slave latches and one error detecting master latch, Cut2 needs three slave latches and one non-error detecting master latch. Suppose the area cost of an EDL is three units, and the cost of a slave, or a non-EDL master latch is one. Cut1 needs 5 units cost, but Cut2 only needs 4 units cost. Traditional min-area re-timing tries to minimize the number of latches which is Cut1 in this example, our goal is to find Cut2.

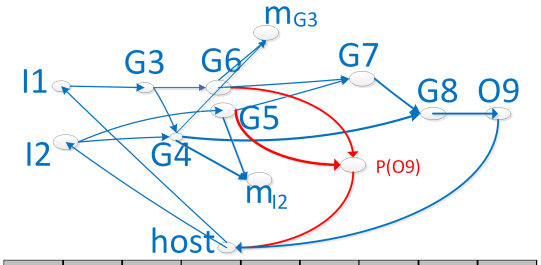
3.2 The Approach

Our approach to solve this problem is to enhance the traditional latch-based retiming algorithm to understand timing resilient designs. To do this, for each target master t, we pre-compute the cut set of gates $g(t)$ for which if the slave latches are moved forward beyond these gates, then the target master t need not be error detecting. This happens if the retiming of latches beyond the gates in $g(t)$ satisfies (5) $< \Pi$. In particular, for a master t,



GATE(v)	d	$D^f(v)$	$D^b(v,O7)$	Region	B(v)	X(v)	Node(v)	B(v)	X(v)
I1	0	0	9	Vm	0	0	host	2	-2-c
I2	0	0	7	Vr	0	0	P(O9)	0	c
G3	2	2	7	Vr	0	0	mG3	-1	1
G4	2	4	1	Vr	1	-1	mI2	-1	1
G5	5	5	2	Vr	1	-1			
G6	5	7	2	Vr	1	-1			
G7	1	8	1	Vn	-1	1			
G8	1	9	0	Vn	-1	1			
O9	0	9	0	Vn	-1	1			

Figure 4: An illustrative circuit.



edge(u,v)	$\beta(e_{uv})$	$w(e_{uv})$	edge(u,v)	$\beta(e_{uv})$	$w(e_{uv})$	edge(u,v)	$\beta(e_{uv})$	$w(e_{uv})$
(I1,G3)	1	0	(G6,mG3)	0.5	0	(G7,G8)	1	0
(I2,G4)	0.5	0	(G4,mG3)	0.5	0	(G8,O9)	1	0
(I2,G5)	0.5	0	(G3,G4)	0.5	0	(O9,h)	0	0
(h,I1)	1	1	(G3,G6)	0.5	0	edge(u,v)	cost	$w(e_{uv})$
(h,I2)	1	1	(G6,G7)	1	0	G5,P(O9)	0	0
(G4,mI2)	0.5	0	(G5,G7)	1	0	G6,P(O9)	0	0
(G5,mI2)	0.5	0	(G4,G8)	1	0	(P(O9),h)	-c	0

Figure 5: Node graph representation of the circuit in Fig. 4.

the corresponding $g(t)$ is the set of gates:

$$g(t) = \{v | \exists n \in FO(v), A(v,n,t) \leq \Pi \\ \wedge \exists k \in FI(v), A(k,v,t) > \Pi\} \quad (8)$$

Notice that if the longest combinational logic path delay from the previous master stage s to the master latch t in the subsequent stage is larger than Π, then $g(t)$ is the empty set and master t must be error detecting regardless of where the slave latches are re-timed. If the combinational logic delay from any previous master stage s to master t is smaller than $\phi_2+\gamma_2$, then master t is an non-error detecting latch regardless of where the slave latches are re-timed. $g(t)$ is also empty in this case. If $g(t)$ is non-empty, every path from the previous stage to the master t passes at least one gate in $g(t)$. If all the slave latches are placed in the fan-out cone of $g(t)$, and in the fan-in cone of t, master t may be a (relatively low cost) non-error detecting latch since any slave latches placed on edge $e(u,v)$ in the fan-out cone of $g(t)$ satisfy $A(u,v,t) \leq \Pi$. However, if any slave latch is positioned between gates u,v in the fan-in cone of $g(t)$, t is forced to be an error-detecting latch since $A(u,v,t) > \Pi$. To find $g(t)$, we do a depth first search (DFS) backwardly from master t until the constraints of $g(t)$ are satisfied. The $g(O9)$ of the re-timing example in Fig. 4 is $\{G6, G5\}$, since $A(G6,G7,O9) = 9 < 10, A(G3,G6,O9) = 12 > 10$ and $A(G5,G7,O9) = 7 < 10, A(I2,G5,O9) = 12 > 10$. If slave latches are placed at the fan-out of $G6$ and $G5$, latch $O9$ can be non-EDL. Cut2 satisfies this requirement, but Cut1 does not.

Our approach is to create a retiming graph with extra edges and nodes to reflect the benefits of moving latches to the point where specific masters need not being error detecting. In particular, for each target master latch t, we add edges from all gates in $g(t)$ with cost 0 to a pseudo node $P(t)$. We connect the pseudo node $P(t)$ to the host node h with another edge with negative cost $-c$, where c is the overhead of an error detecting latch. We define the set of additional edges added to traditional retiming graph as $E2$. The original edges of the retiming graph are in set $E1$. In addition, we place the nodes of the traditional retiming graph in set $V1$, including the host node h, and the pseudo nodes $P(t)$ in set $V2$, as illustrated in Fig. 5. The nodes m_{G3}, m_{I2} in Fig. 5 are pseudo nodes used to model fanout sharing as described in [12]. $E1, V1$ are shown in blue in Fig. 5, while $E2, V2$ are shown in red. Notice that we create a pseudo node $P(O9)$ for master latch $O9$, and $G5, G6 \in g(O9)$ are connected to $P(O9)$. If a slave latch is placed on $edge(u,v) \in E1$, we have the slave latch cost $\beta(e_{uv})$ as the table in Fig. 5 indicates. If a slave latch is placed on edge $(P(O9), h) \in E2$, we know that the slave latches are moved to the fan-out of $g(O9)$ so that $O9$ can be non-error-detecting and we add cost $-c$ to the objective function to deduct the associated EDL overhead. Lastly, we list in Fig 5 the original number of slave latches on each edge before retiming $w(e_{uv})$. Initially the slave latches are at the inputs, so $w(e_{h,I1}), w(e_{h,I2})$ are 1 and $w(e_{uv})$ of all other edges are zero.

3.3 Retiming Regions

Since we only retime slave latches between master stages, and the slave latches before retiming are placed at the inputs of combinational logic, as illustrated in Fig. 3(a), the retiming value $r(v)$ can only be -1 or 0. To reduce the complexity of the retiming algorithm, we pre-divide the graph nodes into three regions.

1. V_m is the set of gates v for which there exists a terminating latch t for which $D^b(v,t) > \phi_2+\gamma_2+\phi_1$. The slave latches should be retimed through these gates (i.e., $r(v) = -1$) and thus after retiming there should be no slave latches in this region. Otherwise, Constraint (7) is violated. For example, for the circuit shown in Fig. 4, V_m is $\{I1\}$ as $D^b(I1, O9) = 9$ which exceeds $\phi_2+\gamma_2+\phi_1 = 7.5$.
2. V_n is the set of gates v whose $D^f(v)$, the delay from a master s to gate v, exceeds $\phi_1+\gamma_1+\phi_2$. These gates can be determined by analyzing $D^f(v)$ via static timing analysis. No slave latches should be retimed through such gates (i.e., $r(v) = 0$). Otherwise, Constraint (6) is violated. For example, for the circuit shown in Fig. 4, V_n is $\{G7, G8, O9\}$ as $D^f(v)$ of $G7, G8, O9$ equals 8, 9, and 9 which all exceed $\phi_1+\gamma_1+\phi_2 = 7.5$.
3. V_r is the remaining region of combinational gates except V_m, V_n. $\forall v \in V_r, -1 \leq r(v) \leq 0$. This is the region where slave latches can be positioned after retiming. The optimization procedure only needs to determine the value of $r(v)$ in this region. For example, for the circuit shown in Fig. 4, V_r is $\{I2, G3, G4, G5, G6\}$.

3.4 ILP formulation

We can minimize the area of slave latches and error detecting latches with the following Integer Linear Programming (ILP) formulation.

$$\min \sum_{v \in V1} \left[\sum_{\forall u \in FI(v)} \beta(e_{uv}) - \sum_{\forall u \in FO(v)} \beta(e_{vu}) \right] r(v) + \\ \sum_{P(t) \in V2} -c \times (r(h) - r(P(t))) \\ s.t. \quad r(u) - r(v) \leq w_{u,v} \quad \forall (u,v) \in E_1 \\ r(u) - r(v) \leq 0 \quad \forall (u,v) \in E_2 \\ L_v \leq r(v) \leq U_v, r(v) \in \mathbb{Z} \quad \forall v \in V \quad (9)$$

Equation (9) is similar to traditional retiming (4) except for the extra node set $V2$ in the objective function and the extra constraints associated with $E2$. As Fig. 5 shows, if a slave latch is positioned on edge $e_{uv} \in E1$ after re-timing, the cost is $\beta(e_{uv})$. If a slave latch is positioned on edge $e_{P(t),h}$, the cost is $-c$ because the master t can be non-error detecting, saving c units of cost. $r(h) - r(P(t))$ is the number of slave latches on edge $e_{P(t),h}$ after retiming. The $-c \times (r(h) - r(P(t)))$ in the objective function represents the reduction of EDL overhead.

The extra constraints $r(u) - r(v) \leq 0, \forall (u,v) \in E2$ guarantees that there is a slave latch on $e_{P(t),h}$ only when the slave latches are moving beyond all gates in $g(t)$. As discussed in Section 3.3, the bounds on $r(v)$ depend on the region of gate v. After the ILP solver returns the value of the $r(v)$ variables, the slave latches are placed on the edge where $w_{u,v} + r(v) - r(u) = 1$.

To construct the ILP for our example in Fig. 4, we use the retiming graph in Fig. 5. Each $edge(u,v)$ represents constraints $r(u) - r(v) < w(e_{uv})$, $w(e_{uv})$ is represented in the table in Fig. 5. The objective function can be obtained using the listed $\beta(e_{uv})$ values and a proper choice of EDL overhead c. The upper bound and lower bound of each node depends on the region. If $v \in V_m, -1 \leq r(v) \leq -1$. If $v \in V_n, 0 \leq r(v) \leq 0$, i.e., $r(v) = 0$. If $v \in V_r, -1 \leq r(v) \leq 0$. Also, $r(h) = 0$ and nodes in $V2$ have the same bounds nodes as nodes in V_r. The ILP solver would return $r(I1) = r(I2) = r(G3) = r(G4) = r(G5) = r(G6) = r(P(O9)) = -1$ with all other $r()$ values set to 0.

4. NETWORK FLOW FORMULATION

4.1 The Formulation

To solve (9) with integer retiming values $r(u)$, we can, in principle, call an Integer Linear Programing (ILP) solver directly. However, ILP is NP-hard and thus the worst case running time is exponential in the size of the problem statement. Rather, we show that our modified retiming formulation (9) can be mapped to a min cost network flow algorithm, similar to traditional retiming [15], and solved with the network simplex method [1] in polynomial time.

Towards this goal, instead of solving the min in (9), we solve a max of the modified objective function in which each coefficient is multiplied by -1 in the Lagrangian expression equation (11). To make the following Lagrangian function (11) simpler to read, we define

$$B(v) = \sum_{\forall j \in FO(v)} \beta(e_{vj}) - \sum_{\forall j \in FI(v)} \beta(e_{jv}). \tag{10}$$

Adding the constraints in (9) with Lagrange multipliers x_{uv} for each corresponding constraints in $(u,v) \in E1, E2$ of (9) we get the following Lagrange function.

$$\begin{aligned} L(r,x) = & \sum_{v \in V1} r(v)[B(v)] + \sum_{P(t) \in V2} r(P(t))[-c] + r(h)c|V2| \\ & + \sum_{(u,v) \in E1} (w_{uv} + r(v) - r(u))\, x_{uv} \\ & + \sum_{P(t) \in V2} \sum_{g \in g(t)} (r(P(t)) - r(g))\, x_{g,P(t)} \\ & + \sum_{P(t) \in V2} (r(h) - r(P(t)))x_{P(t),h} \end{aligned} \tag{11}$$

We next introduce the notion of the demand of node v,

$$X(v) = \sum_{(u,v) \in E} x_{uv} - \sum_{(v,u) \in E} x_{vu}, \tag{12}$$

i.e., the amount of total flow on v's inputs minus the total flows on v's outputs. We can then transform (11) into a min cost network flow formulation similar to traditional min area retiming [12, 15], with the difference that we have extra pseudo nodes $P(t)$ for each master latch t:

$$\begin{aligned} & \min \sum_{(u,v) \in E1} w_{uv} x_{uv} \\ s.t. \quad & X(v) = -B(v) && \forall v \in V1 - h \\ & X(P(t)) = c && \forall P(t) \in V2 \\ & X(h) = -B(h) - c \times |V2| \\ & x_{u,v} \geq 0 && \forall u,v \in V1 \cup V2 \end{aligned} \tag{13}$$

This is the dual problem of (9) and as such the $r(v)'s$ in (11) are now the implicit Lagrange multipliers of this formulation. The physical meaning of x_{uv} is the amount of flow on edge $e(u,v)$ and this flow has a cost w_{uv} for each unit flow on $e(u,v)$. The constraints set the demand of each type of node in the retiming graph and ensure the amount of flow on each edge is non-negative. For example, the specific $B(v)$ and $X(v)$ values of each node in our sample circuit are shown in Fig. 4. Each $e(u,v)$ in Fig. 5 represents the variable x_{uv}. The constraint is that the total flow in to a node minus the total flow out of a node must equal its demand. Using node $G3$ as an example, $x_{I1,G3} - x_{G3,G4} - x_{G3,G6} = -B(G3) = 0$. Given the demand of each node, and flow cost of each edge, the Network Simplex Solver return its corresponding flow x_{uv} of each edge, and the potential $r(v)$ of each node.

Notice that the constraint that $r(v)$ in (9) need be integer is relaxed because the network simplex algorithm [1] guarantees $r(v)$ will be integral when $w_{u,v}$ are integral. Also, the upper bound and lower bound of each $r(v)$ is not shown in min cost network flow problem because we use the trick proposed in [15] to add extra edges to the host node to capture the upper and lower bounds.

5. EXPERIMENTAL RESULTS

In this section, we present the results of our experiments to evaluate the proposed retiming algorithm. The experiments are conducted on a Unix machine with a 2.7 GHz CPU and 8 GB RAM. We select benchmark circuits from ISCAS89, MCNC, and other large industrial benchmarks obtained from [2] and used the *NanGate* 45nm Open Cell Library for gate-level synthesis. The benchmark circuits are all flip-flop based designs. To generate two-phase latch-based designs we split each flip-flop into a master and slave latches. As described in Section 3, we keep the master latches fixed and try to reposition the slave latches. $\Pi + \phi_1$ in our experiment is the longest combinational logic path delay between any two master stages obtained through static timing analysis plus D to Q delay of a master latch and a slave latch. We set $\phi_1 = \phi_2 = 0.3\Pi$, and $\gamma_1 = \gamma_2 = 0.2\Pi$ as in [8]. As Section 2.1 describes, master latches with arrival times larger than Π must have EDL.

We read in the synthesized gate-level circuits, generate the min cost network flow graph of (13) with a custom C++ program, and call the *IBM CPLEX Optimizer* to solve the network simplex problem and obtain the optimal cost of the slave and master latches (both error-detecting and non-error-detecting). We compare our total cost to that obtained after traditional non-EDL-aware re-timing which ignores timing resiliency, as was done in [7,8]. For non-EDL-aware re-timing we minimize the number of slave latches using the traditional re-timing algorithm and check which master latches need to have EDL via static timing analysis. Our EDL-aware re-timing, on the other hand, tries to minimize both the cost of master and slave latches simultaneously, considering the cost of the EDL, as described in Section 4.

Table 1 shows the benchmark results comparing our EDL-aware re-timing to that of non-EDL-aware re-timing. The second column, # of S, is the number of master latches in each benchmark circuit. The third column, # of C, is the number of combinational gates. The cost of each slave and non-error detecting latch is 1 unit and the cost of an error detecting latch is $c + 1$. The total cost is the summation of costs for all the time borrowing slave latches as well as the error detecting and non-error detecting master latches. In practice, this cost may represent area, clock load, or energy consumption. There are different proposed EDL designs in the literature with different trade-offs in power, area, delay, and robustness. For this reason we report our results with a range of EDL costs, similar to [9]. In particular, for EDL cost overheads of $c = 2, 1$, and 0.5 our algorithm achieves a cost reduction of 19.6%, 12.5%, and 6.6%, respectively. Note that to save the space in the table, we only report the running time of the EDL-aware re-timing. The CPU time of non-EDL-aware re-timing is on average 13% less than that of our EDL-aware algorithm. Even for the large industrial case, *leon3*, our CPU time is less than 11 minutes which shows the efficiency of the approach.

6. CONCLUSIONS

This paper presents a method to modify the traditional retim-

	circuit size		EDL overhead c=2		EDL overhead c=1		EDL overhead c=0.5	
Name	# of S	# of C	cost reduction	CPU-time(s)	cost reduction	CPU-time(s)	cost reduction	CPU-time(s)
frg2	139	1022	15.78%	0.35	8.65%	0.31	4.41%	0.31
i10	364	2724	34.03%	0.67	20.63%	0.61	11.04%	0.60
i11	364	2724	35.90%	0.75	21.83%	0.61	12.24%	0.61
leon2	149577	1119384	3.83%	342.83	2.53%	341.45	1.49%	293.99
leon3	185196	1272597	18.55%	635.98	11.85%	646.69	6.90%	567.06
leon3mp	109089	850387	12.69%	290.20	8.24%	296.48	4.88%	245.88
s1196	32	461	20.00%	0.20	15.32%	0.20	4.61%	0.18
s1238	32	490	16.27%	0.22	5.19%	0.20	4.60%	0.18
s13207	790	4250	36.79%	0.86	22.76%	0.83	12.92%	0.76
s1423	79	625	28.93%	0.22	17.89%	0.21	9.52%	0.21
s1488	25	581	1.89%	0.21	12.35%	0.22	0.73%	0.21
s1494	25	590	1.90%	0.22	1.25%	0.23	5.18%	0.20
s38417	1742	11877	33.10%	2.54	19.84%	2.45	10.90%	2.36
s38584	1730	14335	40.35%	3.08	25.56%	3.16	14.31%	2.64
vga_lcd	17188	155397	41.67%	35.22	27.57%	35.14	12.36%	29.97
netcard	97888	983683	1.45%	474.70	0.84%	456.06	0.44%	458.94
ray	23648	235526	3.92%	60.08	1.56%	58.89	0.87%	62.00
uoft_raytracer_opt	17112	218671	5.45%	54.10	1.94%	51.01	1.01%	51.30
Average			19.6%	105.69	12.5%	105.26	6.6%	91.75

Table 1: Experimental results

ing graph with extra edges and nodes to take the cost of error detecting latches into account for two-phase latch-based resilient circuit design. Instead of solving the objective function directly with an ILP, we show that our problem can be mapped to a min cost network flow problem and be solved with a simplex network solver, similar to traditional min area re-timing. For future work, we plan to extend our model to include more general patterns of time-borrowing and error-detecting stages and consider including a notion of the error-rate into our objective function.

7. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. Network flows: theory, algorithms, and applications. 1993.

[2] Collection of digital design benchmarks. http://ddd.fit.cvut.cz/prj/Benchmarks/.

[3] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S. Lu, T. Karnik, and V. De. Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance. *IEEE JSCC*, 44(1):49–63, Jan 2009.

[4] M. Choudhury, V. Chandra, K. Mohanram, and R. Aitken. Timber: Time borrowing and error relaying for online timing error resilience. In *DATE*, pages 1554–1559, March 2010.

[5] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw. Razor II: In situ error detection and correction for PVT and SER tolerance. *IEEE JSCC*, 44(1):32–48, Jan 2009.

[6] R. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge. Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits. *Proc. of the IEEE*, 98(2):253–266, Feb 2010.

[7] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester. Bubble razor: Eliminating timing margins in an ARM cortex-M3 processor in 45 nm CMOS using architecturally independent error detection and correction. *IEEE JSCC*, 48(1):66–81, Jan 2013.

[8] D. Hand, M. Trevisan Moreira, H.-H. Huang, D. Chen, F. Butzke, Z. Li, M. Gibiluka, M. Breuer, N. Vilar Calazans, and P. Beerel. Blade – a timing violation resilient asynchronous template. In *ASYNC*, pages 21–28, May 2015.

[9] H.-H. Huang, H. Cheng, C. C. Chu, and P. A. Beerel. Area optimization of resilient designs guided by a mixed integer geometric program. In *DAC*, June 2016.

[10] A. B. Kahng, S. Kang, J. Li, and J. Pineda De Gyvez. An improved methodology for resilient design implementation. *TODAES*, 20(4):66, 2015.

[11] S. Kim and M. Seok. Variation-tolerant, ultra-low-voltage μP with a low-overhead, within-a-cycle in-situ timing-error detection and correction technique. *IEEE JSSC*, 50(6):1478–1490, Jun 2015.

[12] C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. Technical Report D-SRC-13, Digital (Palo Alto, CA US ; Cambridge, MA US). Systems research center, 1986.

[13] Y. Liu, R. Ye, F. Yuan, R. Kumar, and Q. Xu. On logic synthesis for timing speculation. In *ICCAD*, pages 591–596. IEEE, 2012.

[14] Y. Liu, F. Yuan, and Q. Xu. Re-synthesis for cost-efficient circuit-level timing speculation. In *DAC*, pages 158–163. ACM, 2011.

[15] N. Maheshwari and S. Sapatnekar. Efficient retiming of large circuits. *IEEE Trans. on VLSI*, 6(1):74–83, 1998.

[16] N. Maheshwari and S. S. Sapatnekar. Efficient minarea retiming of large level-clocked circuits. In *DATE*, pages 840–845, Feb 1998.

[17] B. Munger, D. Akeson, S. Arekapudi, T. Burd, H. R. Fair, J. Farrell, D. Johnson, G. Krishnan, H. McIntyre, E. McLellan, et al. Carrizo: A high performance, energy efficient 28 nm APU. *IEEE JSCC*, 51(1):105–116, 2016.

[18] M. C. Papaefthymiou and K. H. Randall. Tim: A timing package for two-phase, level-clocked circuitry. In *DAC*, pages 497–502. ACM, 1993.

[19] S. S. Sapatnekar and R. B. Deokar. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE Trans. on CAD*, 15(10):1237–1248, 1996.

[20] S. S. Sapatnekar and R. B. Deokar. Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits. *IEEE Trans. on CAD*, 15(10):1237–1248, 1996.

[21] V. Singhal, S. Malik, and R. K. Brayton. The case for retiming with explicit reset circuitry. In *ICCAD*, pages 618–625, 1997.

[22] J. Tschanz, K. Bowman, S. Walstra, M. Agostinelli, T. Karnik, and V. De. Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance. In *VLSI Circuits*, pages 112–113, June 2009.

[23] R. Ye, F. Yuan, H. Zhou, and Q. Xu. Clock skew scheduling for timing speculation. In *DATE*, pages 929–934. IEEE, 2012.