Retiming of Two-Phase Latch-Based Resilient Circuits

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. We compare our approach to that obtained by traditional min area re-timing which ignores the resiliency overheads on a wide variety of benchmarks. Our experimental results show that our method reduces the cost of the sequential elements by an average of up to 20%. The solutions are found within 15 minutes even for large industrial circuits, demonstrating the computational efficiency of the approach.

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 process, voltage, and temperature 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 resilient circuits have been proposed. For example, canary FFs predict when the design is close to a timing failure [12, 25]. Designs can then adjust their supply voltage or clock frequency either statically or dynamically to ensure correct operation at the edge of failure. 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, 14, 18, 26]. Liu et al. [14] proposed to reshape the delay distribution of near-critical paths to reduce timing errors. Dynatune [18] optimizes throughput by selectively choosing low V_{th} gates to speed up near-critical paths. Ye et al. [26] and Kahng et al. [10] both use clock skew scheduling to reduce timing errors.

Of particular importance to this paper 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 error detecting logic 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 re-running logic synthesis with a tighter max delay constraint to reduce the amount of EDL needed at the cost of increased logic area.

The closest to our work is that proposed by [15] 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 number of 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 allow time-borrowing 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 [23, 24].

We propose an extension of the traditional retiming al-

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

gorithm that models the cost of the error-detecting latches and 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 formulation that solves the modified retiming problem and then transform it into a network flow problem. The network flow reformulation is an exact, efficient solution to the problem. 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 compute time.

The remainder of the paper is organized as follows. Section 2 introduces the background of retiming and resilient circuits. Section 3 describes the model of our problem 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. RETIMING AND RESILIENT CIRCUIT BACKGROUND

2.1 Two-Phase Latch-Based Resilient Circuits

The clock model [19] of latch-based circuit design often assumes k-phases with k periodic clock signals, $\Pi = \langle \phi_1, \gamma_1, \phi_2, \gamma_2, \dots, \phi_k, \gamma_k \rangle$, 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 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 consider symmetric clock scheme for resilient latch-based design, $\phi_1 = \phi_2$ and $\gamma_1 = \gamma_2$. The resiliency window [8, 11] is ϕ_1. Fig. 1 shows the clock diagram of a two-phase latch-based resilient circuit. 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. Time borrowing is allowed for slave stage and only master stage can be error detecting. This reduces the overhead of EDL while simultaneously reducing the complexity of the timing constraints [8, 11].

2.2 Error

2.3 Min-Area Retiming

There are two traditional objectives for retiming algorithms, minimizing the clock period [13, 20] and minimizing the number of sequential elements [13, 16, 17, 22]. We now review the traditional min-area re-timing algorithm because it is more closely related to our work. As in [13], 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 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. D_{uv} and W_{uv} are defined as follows.

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

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

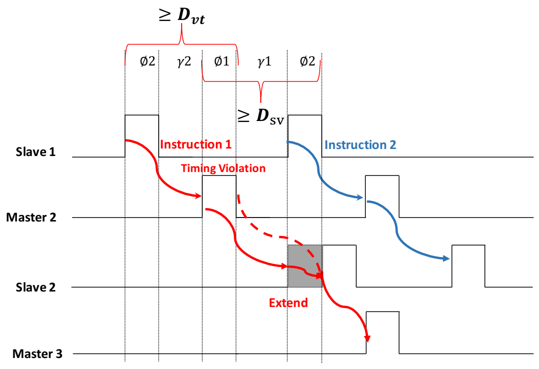


Figure 1: Timing diagram of a resilient circuit.

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 between gate u, v after retiming. The objective function and constraints of the classic retiming algorithm [13] for flip-flop based design are as follows.

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

P in (3) is the cycle time of clock. The path delay between any two flops should be less than P to avoid violating setup timing constraints. Suppose the fanout of gate u is k, $\beta(e_{uv}) = \frac{1}{k}$. $\beta(e_{uv})$ is the cost coefficient of a latch or flip-flop on edge e_{uv} that models fanout sharing with the introduction of pseudo nodes [13]. To simplify the constraints, [16] re-writes them as follows.

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

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 [16]. The values of L_u and U_u are calculated through ASAP and ALAP skews [21]. 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. PROBLEM STATEMENT AND 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. The original position of slave latches before retiming are at the output of their associated master latches. The slave latches become transparent at a fixed time $\phi_1 + \gamma_1$ and remain transparent until time $\phi_1 + \gamma_1 + \phi_2$. Thus, slave latches do not start propagating signals through the subsequent combinational logic until time

$\phi_1 + \gamma_1$, as shown in Fig. 1. The total delay from a master s through a repositioned slave latch v to a given master t is thus

$$\max\{\phi_1 + \gamma_1, D^f(v)\} + D^b(v,t) + D_l \tag{5}$$

Here, $D^f(v)$ is the maximum delay from any master latch in the current stage to the slave latch at output of gate v and can be 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 master latch t in the subsequent stage. It is the delay calculated backwardly starting from master t to slave v. D_l is the D to Q delay of the slave latch.

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 arrival times 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, *target* master latches.

Note, however, there is 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

$$D^f(v) \le \phi_1 + \gamma_1 + \phi_2. \tag{6}$$

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,

$$D^b(v,t) \le \phi_2 + \gamma_2 + \phi_1 \tag{7}$$

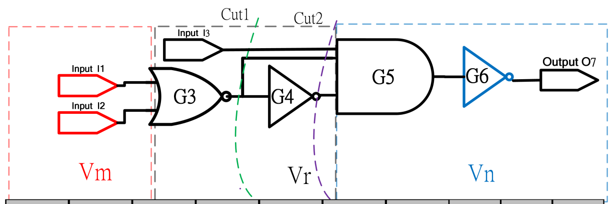
The goal of our work is to re-time the slave latches to minimize the cost of the sequential logic, including the total number of slave and master latches and the error-detecting overhead, while still satisfying the above constraints.

3.2 The Approach

Our approach is to enhance the traditional retiming algorithm to understand resilient designs. To do this, for each target master t, we pre-compute the 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 $g(t)$ satisfies (5) $< \Pi$. In particular, for a master t, the corresponding $g(t)$ is the set of nodes:

$$\begin{aligned} g(t) = \{v \mid & \max\{\phi_1 + \gamma_1, D^f(v)\} + D^b(v,t) + D_l \le \Pi \wedge \\ & \exists k \in FI(v), \max\{\phi_1 + \gamma_1, D^f(k)\} + D^b(k,t) + D_l > \Pi\} \end{aligned} \tag{8}$$

Notice that if the longest path delay from previous stage to the master latch t in subsequent stage is larger than Π, then $g(t)$ is the empty set and master t must be error detecting



GATE(v)	d	$D^f(v)$	$D^b(v,O7)$	Region	B(v)	X(v)
I1	0	0	9	Vm	0	0
I2	0	0	9	Vm	0	0
I3	0	0	2	Vr	0	0
G3	2	2	7	Vr	-1	1
G4	5	7	2	Vr	1	-1
G5	1	8	1	Vr	-1	0
G6	1	9	0	Vn	0	0
O7	0	9	0	Vn	-1	1

Node(v)	B(v)	X(v)
host	3	-3-c
P(O7)	0	c
mG3	-1	1

Figure 2: An illustrative circuit, $\phi_1 = \gamma_1 = \phi_2 = \gamma_2 = 2.5$.

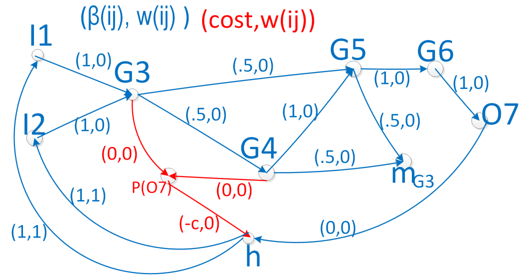


Figure 3: Node graph representation of the circuit in Fig. 2.

regardless of where the slave latches are retimed. If the delay to master t is smaller than $\phi_2 + \gamma_2$, then t is an non-error detecting latch regardless of where the slave latches are retimed. $g(t)$ is also empty in this case. If all the slave latches are at the outputs of gates in $g(t)$, master t may be a (relatively low cost) non-error detecting latch. However, if any one of the slave latches at the output of $g(t)$ is positioned in the fan-in cone of $g(t)$, t is forced to be an error-detecting latch.

Fig. 2 is an example used to illustrate our approach. 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. Slave latches before retiming are at the inputs of the circuit, as illustrated with the purple latches. The master latches open at time 0, and are transparent for ϕ_1. As mentioned earlier, the slave latches open at time $\phi_1 + \gamma_1$ and are transparent for time ϕ_2 as shown in Fig. 1. The initial position of slave latches are at the outputs of master latches s. Assume $\phi_1 = \gamma_1 = \phi_2 = \gamma_2 = 2.5, D_l = 0$, we get $\max\{\phi_1 + \gamma_1, D^f(G3a)\} + D^b(G3a, O7) = 7 \le \Pi = 10$, $\max\{\phi_1 + \gamma_1, D^f(G4)\} + D^b(G4, O7) = 9 \le \Pi = 10$, and $\max\{\phi_1 + \gamma_1, D^f(G3)\} + D^b(G3, O7) = 12 > \Pi = 10$, $g(O7) = \{G3a, G4\}$. If a slave latch is placed at $G5$ or $G6$, timing constraint (6) is violated. As a result, the best position of slave latches are at $G3a$ and $G4$ as shown by the red lines and the master latch $O7$ can be non-error-detecting. However, the traditional min area re-timing method would place the slave latch at $G3$ to minimize the number of slave latches. This leads to $\max\{\phi_1 + \gamma_1, D^f(G3)\} + D^b(G3, O7) = 12 > \Pi = 10$, and $O7$ then must be error detecting. Assume the area of slave latch and the non-error detecting master latch is 1 and the area of error detecting latch is 3. The total cost of traditional re-timing on Fig. 2 is 4 (i.e., 1 slave latches and 1 error detecting master latch), while the total cost of our approach is 3 (i.e., of 2 slave latches and 1 non-error detecting master latch).

We 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. We add edges from all gates in $g(t)$ with cost 0 to a pseudo node $P(t)$ for each master latch 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 retiming graph are in set $E1$. The nodes of traditional retiming graph are in $V1$, including the host node h, and the pseudo nodes $P(t)$ are in $V2$ as shown in Fig. 3. The node m_{G3} in Fig. 3 is a pseudo node to handle fanout sharing as described in [13]. $E1, V1$ are shown in blue in Fig. 3, while $E2, V2$ are in red. If the latches are moved beyond all the gates in $g(t)$, the optimization program will set $r(P(t)) = -1$, and add a latch to the pseudo edge $e_{P(t),h}$ to reduce the total cost of the cost function (9). For example, as Fig. 3 shows, if $r(G3) = r(G4) = -1$, there is a latch retimed to the pseudo edges $e_{G3,P(O7)}$ and $e_{G4,P(O7)}$ and the optimizer will set $r(P(O7)) = -1$ to retime them to pseudo edge $e_{O7,h}$ to minimize the overall cost. Note here we do not create extra nodes in the retiming graph to model sharing of latches across fan-outs like $G3a$ and $G3b$. If there is a slave latch on $e_{G3,G5}$, and not on $e_{G3,G4}$ in Fig. 3, the slave latch is on the fanout $G3a$ in Fig. 2. On the other hand, if there slave latches on both $e_{G3,G4}$ and $e_{G3,G5}$ in the retiming graph, the fan-outs of $G3$ share one slave latch in the real circuit latch as denoted by the blue line in Fig. 2.

3.3 Retiming Regions

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, in the circuit shown in Fig. 2, V_m is $\{I1, I2\}$ as $D^b(I1, O7) = D^b(I2, O7) = 9$ 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, in the circuit of Fig. 2, V_n is $\{G5, G6, O7\}$ as $D^f(v)$ of $G5, G6, O7$ 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. $\forall v \in V_r, -1 \le r(v) \le 0$. 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. In the circuit shown in Fig. 2, V_r is $\{G3, G3a, G3b, G4\}$.

3.4 ILP formulation

We can minimize the area of slave latches and error detecting latches with the following ILP formulation.

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

As described in Section 3.2, $V1$ is the node set of traditional re-timing graph $V1 = h \cup V_r \cup V_m \cup V_n$, $V2$ is the set of pseudo nodes $P(t)$ for each master latch t. $E1$ is the edge set of traditional retiming graph, and $E2$ is the edge set connecting pseudo node $P(t)$ to the host node h, and its corresponding gates $g(t)$. 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. 3 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. The total cost of the slave latches and the cost reduction of the master latches that can be non-error detecting is represented in the objective function (9).

The constraints $r(i) - r(j) \le w_{ij}, \forall (i,j) \in E1$, $r(i) - r(j) \le 0, \forall (i,j) \in E2$ are also illustrated in Fig. 3. As discussed above, the bounds on $r(v)$ depend on the region of gate v. $r(h) - r(P(t))$ is the number of slave latch 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. Since $r(h)$ is fixed at 0, if the optimizer wants to reduce the overhead of a certain master t, $r(h) - r(P(t)) = 1$ and $r(P(t)) = -1$. Also, $r(P(t))$ can be -1 only when $\forall u \in g(t), r(u) = -1$ which means the slave latches are all retimed beyond the gates in $g(t)$. For each master t, the cost on edge $e_{g(t),P(t)}$ is 0 because latches on the pseudo edges $e_{g_(t),P(t)}$ are virtual.

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 [16], and solved with a network simplex solver in polynomial time.

Towards this goal, instead of solving min in (9), we solve a max of the modified objective function in which each coefficient is multiplied by -1 in (10). To make the following formula simpler, we define $B(v) = \left[\sum_{\forall j \in FO(v)} \beta(e_{vj}) - \sum_{\forall j \in FI(v)} \beta(e_{jv})\right]$.

$$\max \sum_{v \in V1} r(v)B(v) + \sum_{P(t) \in V2} -c(r(P(t)) - r(h)) \qquad (10)$$

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

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

Re-arranging (11) and introducing the notion of the demand of v, $X(v) = \left(\sum_{(u,v)\in V} x_{uv} - \sum_{(v,u)\in V} x_{vu}\right)$, we obtain:

$$\begin{aligned}L(r,x) &= \sum_{(u,v)\in E1} w_{uv}x_{uv} + \sum_{v\in V1-h} r(v)\,[B(v) + X(v)] \\ &+ \sum_{P(t)\in V2} r(P(t))\,[X(P(t)) - c] + r(h)\,[X(h) + c|V2| + B(h)] \\ & x_{u,v} \geq 0 \quad \forall u,v \in V1 \cup V2\end{aligned} \tag{12}$$

We can then transform this into a min cost network flow formulation:

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

Equation (13) is the dual problem of (9) and as such the $r(v)'s$ in (11) and (12) are now the implicit Lagrange multipliers of (13). The cost on each edge (u,v) of this dual problem is w_{uv}. The demand of each node $X(v)$, $v \in V1, V2, h$, is derived from (11). The demand of each gate node $u \in V1$, $X(u)$ is $-B(v)$, the demand of pseudo terminator node $P(t) \in V2$, $X(P(t))$ is c, and the demand of host node is $-B(h) - c \times |V2|$.

To show that this transformed problem can be solved with the min cost network flow algorithm [1], we show that: *(a)* $\sum_{u\in V1\cup V2} X(u) = 0$; *(b)* there exists at least one feasible solution; and *(c)* there are no negative cycles in the modified retiming graph. For the traditional retiming graph, constraints *(a), (b)*, and *(c)* are satisfied. We show that our retiming graph with extra edges $E2$, and nodes $V2$ also satisfy these constraints. *(a)* The summation of demand of the original network flow (without psuedo nodes $P(t)$) is zero This implies that $\sum_{u\in V1} X(u) = -c \times |V2|$ on our modified network flow graph, and $\sum_{u\in V1} X(u) + \sum_{P(t)\in V2} X(P(t)) = 0$. To show *(b)*, suppose $x'_{h,g(t)}$ is the original flow in the traditional network flow graph. On our modified min cost low graph, set flow $x_{g(t),P(t)} = \frac{c}{|g(t)|}$ and $x_{h,g(t)} = x'_{h,g(t)} + \frac{c}{|g(t)|}$. All other flows are the same as the flow in the traditional retiming graph without the pseudo nodes $P(t)$. This flow assignment satisfies the demand requirement in (13) and is one feasible solution. To show *(c)*, we note that there is no negative cycles in the traditional min area re-timing graph [1] and the cost of newly added edges $w_{g(t),P(t)}$ and $w_{P(t),h}$ add to zero because $r(g(t)) - r(P(t)) \leq 0$ and $r(P(t)) - r(h) \leq 0$ in (9). As a result, there is no negative cycles in our modified network flow graph.

As (13) is a valid min cost network flow problem, the strong duality principle [1] of (10) and (13) implies the optimal solution of (10) and (13) are the same. Moreover, the constraint that $r(v)$ in (9) need be integer is omitted because the network simplex algorithm [1] guarantees $r(v)$ will be integral when $w_{u,v}$ are integral.

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 design. 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 path delay between any two master stages. $\phi_1 = \phi_2 = 0.3\Pi$, and $\gamma_1 = \gamma_2 = 0.2\Pi$ for our experiment.

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. With the result from the network simplex solver, we place the slave latches in the example circuits, and check the timing constraints to decide which master latches need be error detecting and sum up the cost of the slave and master latches (both error-detecting and non-error-detecting). We compare our cost to that obtained with traditional non-EDL-aware re-timing which minimizes the number of slave latches without information regarding which of the master latches need be error detecting, i.e., we only minimize the number of slave latches through min-area re-timing algorithm for non-EDL-aware re-timing, and check which master latches need to be EDL with timing analysis. Our EDL-aware re-timing tries to minimize both the cost of master latches, slave latches at the same time with the help of extra edges with negative cost Section 3.

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 and in particular adopt the range proposed in [9]. In particular, for EDL cost overheads of

	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

$c = 2, 1$, 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. CONCLUSION

This paper presents a method to modify the traditional retiming 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 ILP, we show that our problem can be mapped to a min cost network flow problem and be solved with 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 are including a notion of the average performance of the resilient circuit 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] Y. Kunitake, T. Sato, H. Yasuura, and T. Hayashida. Possibilities to miss predicting timing errors in canary flip-flops. In *MWSCAS*, pages 1–4, Aug 2011.

[13] 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.

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

[15] Y. Liu, F. Yuan, and Q. Xu. Re-synthesis for

cost-efficient circuit-level timing speculation. In *DAC*, pages 158–163. ACM, 2011.

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

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

[18] M. Nakai, S. Akui, K. Seno, T. Meguro, T. Seki, T. Kondo, A. Hashiguchi, H. Kawahara, K. Kumano, and M. Shimura. Dynamic voltage and frequency management for a low-power embedded microprocessor. *IEEE JSCC*, 40(1):28–35, 2005.

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

[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] 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.

[22] N. Shenoy and R. Rudell. Efficient implementation of retiming. In *ICCAD*, pages 226–233, 1994.

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

[24] H. J. Touati and R. K. Brayton. Computing the initial states of retimed circuits. *IEEE Trans. on CAD*, 12(1):157–162, 1993.

[25] 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.

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