Retiming and Time Borrowing: Optimizing High-Performance Pulsed-Latch-Based Circuits

Seonggwan Lee
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Seungwhun Paik
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Youngsoo Shin
Dept. of Electrical Engineering, KAIST
Daejeon 305-701, Korea

ABSTRACT

Pulsed-latches take advantage of both latches in their high performance and flip-flops in their convenience of timing analysis. To minimize the clock period of pulsed-latch-based circuits for a higher performance, a problem of combined retiming and time borrowing is formulated, where the latter is enabled by using a handful of different pulse widths. The problem is first approached by formulating it as an integer linear programming to lay a theoretical foundation. A heuristic approach is proposed, which solves the problem by performing clock skew scheduling for the minimum clock period and gradually converting skew into a combination of retiming and time borrowing. Experiments with 45-nm technology demonstrate that the clock period close to the minimum can be achieved for all benchmark circuits with an average of 1.03× with less use of extra latches compared to the conventional retiming.

Categories and Subject Descriptors: B.6.1 [**Logic Design**]: Design Styles—*Sequential circuits*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*

General Terms: Algorithms, Design

Keywords: Pulsed-latch, sequential circuit, retiming, time borrowing, clock period

1. INTRODUCTION

Latches are widely used in high-performance custom designs such as microprocessors. Application-specific integrated circuit (ASIC) designs, on the other hand, mostly use flip-flops for their convenience of timing analysis, i.e. each combinational block between flip-flops can be analyzed independently. A flip-flop, however, is slower than a latch; the sequencing overhead of a flip-flop is 3 or 4 FO4 delays while that of a latch is 2 [1]. This is unavoidable because a flip-flop is typically constructed by connecting two latches in a master-slave fashion.

A pulsed-latch [2–8] is a latch driven by a brief clock pulse. Since the length of time when the latch is transparent thereby capturing input data is very short, its behavior is very similar to an edge-triggered flip-flop. Therefore, pulsed-latches have advantages of both latches (low sequencing overhead) and flip-flops (convenience of timing analysis). The clock pulse can be either internally generated by latch itself [2, 4], or generated by pulse generators [3, 7]. When pulse generators are used, they deliver clock pulse only to latches located physically very close because the pulse can be easily distorted. Figure 1 illustrates pulsed-latch-based circuits using pulse generators; a normal clock is delivered, via clock distribution network, from a clock source to pulse generators, which then deliver the pulse to the local latches.

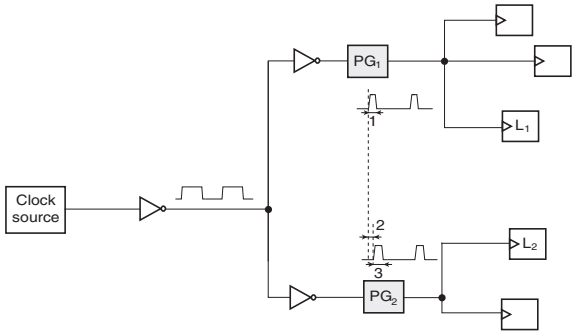

Figure 1: Pulsed-latch-based circuits with local pulse generators PG_1 and PG_2.

To optimize pulsed-latch-based circuits for a higher performance, clock skew scheduling [9] can be applied. It intentionally applies clock skew so that latch-to-latch timing critical paths get more time to compute, which effectively reduces the clock period. Large amount of skew, however, has become difficult to implement, because within-die process variations affect extra buffers and wires that are inserted to realize skew in randomly different amount, thereby causing uncertainties in skew [10, 11]. Small amount of skew, e.g. 10% of the clock period [10], on the other hand, severely restricts the extent of the clock period that can be reduced by clock skew scheduling. This can be alleviated by using time borrowing, which is offered by employing different pulse widths, together with skew [12]. As shown in Figure 1, if L_1 receives a pulse with width 1 and L_2 receives a pulse with width 3, which is also skewed by 2, the combinational logic between L_1 and L_2 can use $(3-1)+2=4$ more time than the clock period; equivalently, for a given delay of combinational logic, the clock period can be reduced. Using a wider pulse, however, comes at the cost of increasing chance of hold time violations.

Due to increasing within-die process variations with technology scaling [13], even small amount of skew is becoming

ICCAD'09, November 2–5, 2009, San Jose, California, USA.

difficult to realize in a reliable manner. As an example, we took a 4-level buffered clock tree and implemented skew of 76 ps in 0.99-V, 45-nm commercial technology. We then performed the Monte Carlo simulation 5000 times to see the quantitative effect of within-die process variations on skew; a standard deviation was 15 ps, which is 20% of a target skew, with the minimum and the maximum skew of 24 ps and 126 ps, respectively.

Retiming [14] is another candidate of optimizing performance. It moves positions of latches across nearby combinational gates, thus is a local optimization as opposed to clock skew scheduling that handles a global clock tree. It therefore is less susceptible to process variations and can overcome the aforementioned limitation of clock skew scheduling. The main limitation of retiming, however, is a large increase in the number of latches. It is commonly observed that the number of latches (or flip-flops) gets doubled or tripled [15].

In this paper, we address a problem of minimizing the clock period of pulsed-latch-based circuits by using retiming and time borrowing, where the latter is realized by employing multiple pulse widths (Section 2). We first solve the problem in optimal way by formulating it as an integer linear programming (Section 3); this, however, is only applicable to circuits of very small size. We thus propose a heuristic, which relies on the equivalence of retiming and clock skew. We initially perform clock skew scheduling to minimize the clock period; skew is then gradually converted to the combination of retiming and time borrowing; the remaining skews after conversion are removed to determine the final clock period (Section 4). Experiments with 45-nm commercial technology shows that the heuristic approach yields clock period close to minimum (average of 1.03×) with 16% increase of latches, which is in contrast to the conventional retiming with the average clock period of 1.11× with 27% increase of latches (Section 5).

2. PROBLEM FORMULATION

In this section, we briefly review conventional retiming for minimum clock period (Section 2.1), which we extend to retiming and time borrowing of pulsed-latches (Section 2.2).

2.1 Retiming

A sequential circuit is modeled as a directed graph $G = (V, E, d, w)$, where $v \in V$ corresponds to a combinational gate with propagation delay $d(v)$, and $e_{uv} \in E$ models a connection from u to v with the number of registers on it denoted by $w(e_{uv})$. A retiming refers to assigning integer to each vertex; the weight of e_{uv} after retiming, denoted by $w_r(e_{uv})$, is defined by

$$w_r(e_{uv}) = w(e_{uv}) + r(v) - r(u), \tag{1}$$

where $r(v)$ is a retiming on v and indicates the number of registers that are moved from its output to its inputs. Any retiming has to satisfy $w_r(e_{uv}) \geq 0$:

$$r(u) - r(v) \leq w(e_{uv}), \quad \forall e_{uv} \in E \tag{2}$$

which is called a condition for a legal retiming.

Retiming problem to minimize the clock period can be stated by

Problem 1 Given $G = (V, E, d, w)$, the retiming problem is to find a legal retiming $r : V \rightarrow Z$ such that the clock period $T = \max_{l:w_r(l)=0} d(l)$ is minimized.

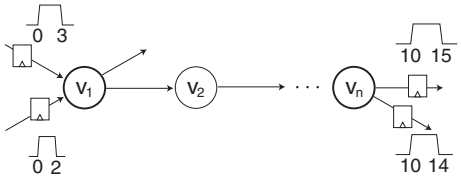

Figure 2: Path delay with time borrowing.

A path of $v_1, v_2, \ldots, v_n$ is denoted by l; thus, if the sum of weights of edges within the path after retiming, denoted by $w_r(l)$, is 0, l is a combinational path. Path delay $d(l)$ is defined by

$$d(l) = \sum_{i=1}^{n} d(v_i). \tag{3}$$

For a particular clock period ϕ, retiming has to satisfy

$$r(u) - r(v) \leq W(u, v) - 1, \quad \forall D(u, v) > \phi \tag{4}$$

where $W(u, v)$ indicates the minimum number of registers on the paths from u to v, and $D(u, v)$ denotes the maximum delay of the paths with $W(u, v)$ registers,[1] i.e. $D(u, v)$ represents the tightest timing requirement between u and v. A system of difference constraints (2) and (4) can be represented by a constraint graph, which then can be solved by Bellman-Ford method [16]. An alternative algorithm with less time complexity exists [17], which does not use D and W. To solve Problem 1, the process of setting up difference constraints and solving them is repeated while we vary ϕ through a binary search.

2.2 Retiming Pulsed-Latches

For retiming and time borrowing of pulsed-latches (retiming pulsed-latches, for brevity), a sequential circuit is modeled as $G = (V, E, d, w, p)$, where $p \in \mathcal{P} \cup \{0\}$ is assigned to each edge $e \in E$ indicating the pulse width of $w(e)$ latches on it; $\mathcal{P}$ is a list of pulse widths available in a library of pulse generators. If $w(e) = 0$, $p(e) = 0$ is presumed. We now state the problem of retiming pulsed-latches.

Problem 2 Given $G = (V, E, d, w, p)$, the problem of retiming pulsed-latches is to find a legal retiming $r : V \rightarrow Z$ and pulse width assignment $p : E \rightarrow \mathcal{P} \cup \{0\}$ such that the clock period $T = \max_{l:w_r(l)=0} d(l)$ is minimized.

Path delay (3) has to be refined to incorporate the amount of time borrowing due to the use of different pulse widths:

$$d(l) = \sum_{i=1}^{n} d(v_i) - \left(\min_{\epsilon \in out(v_n)} p(\epsilon) - \max_{e \in in(v_1)} p(e) \right), \tag{5}$$

where $out(v_n)$ is a set of outgoing edges of v_n and $in(v_1)$ is a set of incoming edges of v_1. Figure 2 shows an example. There are two incoming edges of v_1, each edge with a latch; if there are no clock skews, time 3 is the latest time (ignoring setup time for simplicity) data is available to v_1. We assume two outgoing edges of v_n; time 14 is the latest time data from v_n has to arrive so that it can be safely captured by the latches. The amount of time borrowing by path $v_1, v_2, \ldots, v_n$ is thus $-3 + 4 = 1$; for clock period of 10, for

[1]For any path from u to v with its path delay larger than ϕ, we require at least one register on the path after retiming, i.e. $W(u, v) + r(v) - r(u) \geq 1$, which yields (4).

example, $\sum d(v_i)$ can be as much as 11. Equivalently, when we consider $d(l)$ to determine clock period, the amount of time borrowing has to be subtracted from $\sum d(v_i)$ as in (5).

3. ILP FOR RETIMING PULSED-LATCHES

Problem 2 can be solved by using an integer linear programming (ILP), which we discuss in this section. For the ILP formulation, we have (unknown) variables $r(v)$ for all vertices, which are integers, and $p(e)$ for all edges, which take one of integers from $\mathcal{P} \cup \{0\}$; the objective is to minimize ϕ, which we consider as an integer variable without loss of generality.

3.1 Constraints

We consider two constraints for the ILP formulation. To derive the first constraint, we note that, for a clock period ϕ, retiming has to satisfy

$$W(u,v)+r(v)-r(u) \geq 1, \text{ if } D(u,v)+p(e)-p(\epsilon)-\phi > 0, \quad (6)$$

for all pairs of e and ϵ, where $e \in in(u)$ and $\epsilon \in out(v)$. Note that $W(u,v)+r(v)-r(u)$ is the minimum number of latches on the paths from u to v after retiming (recall (4)), which we denote as $W_r(u,v)$; $D(u,v)+p(e)-p(\epsilon)-\phi$ corresponds to the maximum delay on the paths incorporating time borrowing (recall (5)) subtracted by the clock period, i.e. the amount of timing violation beyond clock period, which we denote as $V(e,\epsilon)$. Condition (6) can be re-written as

$$W_r(u,v) \geq 1, \text{ if } V(e,\epsilon) > 0, \quad (7)$$

or, equivalently, as

$$V(e,\epsilon) \leq 0 \quad \vee \quad W_r(u,v) \geq 1. \quad (8)$$

To convert (8) to a linear form, we define the upper bound of $V(e,\epsilon)$, which is given by

$$\begin{aligned} V_{ub} &= \max_{\forall u,v} D(u,v) + \max_{i \in \mathcal{P} \cup \{0\}} i - \min_{i \in \mathcal{P} \cup \{0\}} i. \quad (9) \\ &= \max_{\forall u,v} D(u,v) + \max_{i \in \mathcal{P}} i. \end{aligned}$$

Note that V_{ub} is a positive constant. We now decompose (8) into two cases: $V(e,\epsilon) \leq 0$ and $V(e,\epsilon) > 0$. If $V(e,\epsilon) \leq 0$, then $W_r(u,v)$ may take any value, except that the value has to be non-negative for a legal retiming, i.e. $W_r(u,v) \geq 0$; this is equivalent to

$$W_r(u,v) \geq \frac{V(e,\epsilon)}{V_{ub}}, \quad (10)$$

because the right-hand side satisfies $-1 < V(e,\epsilon)/V_{ub} \leq 0$ and $W_r(u,v)$ is an integer. If $V(e,\epsilon) > 0$, then $W_r(u,v) \geq 1$ has to be satisfied, which is also equivalent to (10) because the right-hand side, this time, satisfies $0 < V(e,\epsilon)/V_{ub} \leq 1$. Hence, (8) is equivalent to (10), which is now linear and constitutes the first constraint of the ILP formulation.

For the second constraint, if $p(e) > 0$ for any edge e, we have to have at least one latch, i.e. $w_r(e) \geq 1$:

$$w_r(e) \geq 1, \text{ if } p(e) > 0. \quad (11)$$

We use a similar technique ((7) to (10)) to convert (11) to a linear form, which yields

$$w_r(e) \geq \frac{p(e)}{\max_{i \in \mathcal{P}} i}. \quad (12)$$

Note that (12) subsumes (2), i.e. guarantees a legal retiming, because the right-hand side is non-negative.

3.2 ILP Formulation

The ILP to solve Problem 2 can now be summarized by:

$$\begin{aligned} \text{minimize} \quad & \phi \\ \text{subject to} \quad & w(e_{uv}) + r(v) - r(u) \geq \frac{p(e_{uv})}{\max_{j \in \mathcal{P}} j}, \quad \forall e_{uv} \in E \\ & W(u,v) + r(v) - r(u) \geq \frac{D(u,v)+p(e)-p(\epsilon)-\phi}{V_{ub}}. \\ & \forall e, \epsilon \in E : e \in in(u), \epsilon \in out(v) \end{aligned}$$

4. HEURISTIC APPROACH TO RETIMING PULSED-LATCHES

The ILP to solve Problem 2 in the previous section can be applied only to circuits of very small size. In this section, we address a heuristic approach, which utilizes the equivalence of retiming and skew [15]; it consists of three steps:

1. Perform clock skew scheduling to minimize the clock period.
2. Convert skew to retiming and time borrowing.
3. Remove remaining skews, fix hold time violations, and determine the final clock period.

4.1 Clock Skew Scheduling for Minimum Clock Period

This is done by iterative relaxation based approach [18], which is proved to be optimal. For a particular clock period ϕ, setup time constraint is constructed for each pair of latches i and j:

$$S_j \geq S_i + (T_{dq} + D_{ij} - \phi), \quad (13)$$

where S_i denotes a skew, which is initially set to 0; T_{dq} denotes the data-to-Q delay of latch and D_{ij} corresponds to the maximum delay of combinational block between i and j. We take any unsatisfied constraint and set S_j to $S_i + (T_{dq} + D_{ij} - \phi)$, i.e. force left- and right-hand sides to be equal. If S_j becomes larger than ϕ, process finishes with fail because ϕ cannot be satisfied; otherwise, we take another unsatisfied constraint and repeat the process until all constraints get satisfied, when we return with success.

The minimum clock period can be found by iteratively set ϕ (through a binary search, initially using the minimum and the maximum value of ϕ [15]) and perform the aforementioned clock skew scheduling. Note that, in this approach, hold time constraints are ignored; logic paths that violate hold time constraints are fixed later by introducing extra buffers, which is also common in many approaches to clock skew scheduling [15, 19, 20].

4.2 Convert Clock Skew to Retiming and Time Borrowing

Pulse widths of all the latches are initially set to the minimum value $\rho = \min_{x \in \mathcal{P}} x$. We take a latch i having nonzero skew S_i, and try to reduce its magnitude $|S_i|$ as much as possible by converting it to retiming as explained in Section 4.2.1. If there exists a retiming that decreases $|S_i|$, that retiming becomes a candidate. We pick the next wider pulse width $P_i \in \mathcal{P}$, which implies time borrowing by the amount of $P_i - \rho$. This causes a decrease of skew from S_i to $S_i - (P_i - \rho)$. We then try to reduce the magnitude of new skew by converting it to retiming and decide whether that retiming can be another candidate. The process is repeated for all pulse widths in $\mathcal{P}$; out of all candidates, we select

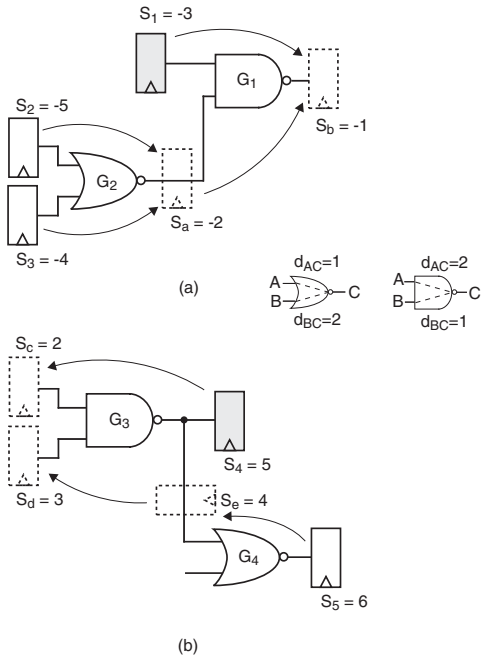

Figure 3: Converting (a) negative skew S_1 to retiming and (b) positive skew S_4 to retiming.

the one (P_i and retiming) that yields the largest decrease of $|S_i|$; if there is a tie, the one with less increase of latches is selected.

4.2.1 Clock Skew to Retiming

We use an algorithm similar to [15] to convert skew to retiming, except that we accept a conversion that does not increase the clock period after retiming move, since relocating latches may affect the delay of some gates. We informally explain the algorithm by using an example in Figure 3. We take a latch 1 having a negative skew $S_1 = -3$ as shown in Figure 3(a). To reduce S_1, we have to perform negative retiming, i.e. move 1 from input of G_1 to its output. This requires a latch at the other input of G_1, which, in turn, is made possible by performing negative retiming on G_2. We thus compute S_a, the skew of the latch if retiming were to be performed on G_2. We finally compute S_b, the skew of the latch after retiming on G_1. If $|S_b| < |S_1|$, which is the case in the example, retiming is performed; otherwise, the latches remain in their original positions.

In Figure 3(b), we take a latch 4 having a positive skew $S_4 = 5$ and try to perform positive retiming to reduce it. This requires a latch at the other fanout of G_3, which, in turn, needs retiming on G_4. We compute S_e, take a minimum of S_4 and S_e, and perform retiming on G_3, which yields S_c and S_d. If both $|S_c|$ and $|S_d|$ are smaller than $|S_4|$, retiming is performed.

4.2.2 Example

We use an example in Figure 4 to illustrate the procedure in this section. Let a list of pulse widths be $\mathcal{P} = \{100, 150\}$ and the pulse width of all latches in a circuit be initialized to 100. We take a latch 1 having a positive skew of $S_1 = 35$ and perform retiming and timing borrowing to reduce it. We look for a candidate location of retiming first with $P_1 = 100$,

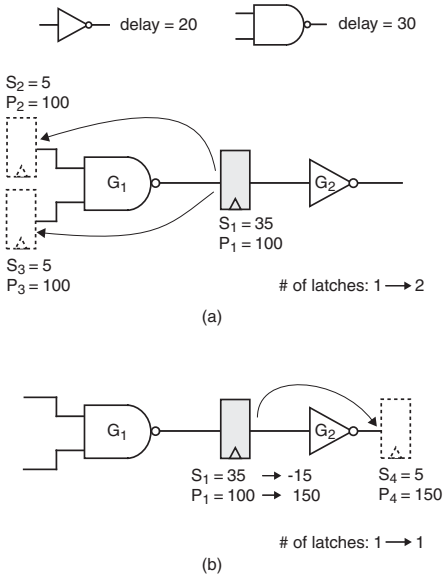

Figure 4: Converting skew S_1 to retiming: (a) when time borrowing is 0 ($P_1 = 100$) and (b) when time borrowing is 50 ($P_1 = 150$).

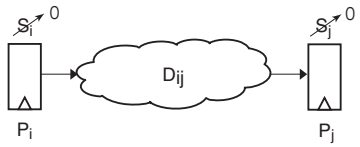

Figure 5: Remove remaining skew and determine new clock period.

which causes relocating latch 1 from the output of G_1 to its inputs as shown in Figure 4(a). This yields two latches while skew is reduced to $S_2 = S_3 = 5$. We then try the next pulse width, $P_1 = 150$. As shown in Figure 4(b), this allows us to use time borrowing of $150 - 100 = 50$, which, in turn, lets S_1 be decreased from 35 to -15. Since the skew is now negative, retiming is performed in the opposite direction, i.e. from the input of G_2 to its output. This yields the same number of latch while skew is reduced to $S_4 = 5$. Since the skews are the same in both retiming moves in this particular example, we pick Figure 4(b) due to its less number of latches.

Note that the use of both retiming and time borrowing allows retiming to be performed in both positive and negative directions whereas the conventional retiming is performed in only one direction. This is a key benefit of using time borrowing along with retiming.

4.3 Determine Final Clock Period

After we convert skew to mix of retiming and time borrowing, the remaining skews are removed by increasing the clock period. Consider the combinational block between latches i and j, denoted by $i \rightsquigarrow j$, as shown in Figure 5. Its maximum combinational delay is $T_{dq} + D_{ij}$; the amount of time borrowing is $P_j - P_i$, which has to be subtracted from the maximum combinational delay when we decide the clock period; the effect of skew is $S_j - S_i$, which also has to be subtracted from the maximum combinational delay, but we now regard it as zero. Therefore, the clock period with zero

skew is given by

$$T_{0-skew} = \max_{\forall i \rightsquigarrow j} [T_{dq} + D_{ij} - (P_j - P_i)]. \quad (14)$$

Once T_{0-skew} is determined, logic paths that violate hold time constraints are fixed by introducing extra buffers [21]. Note that this may further increase the clock period due to finite number of buffer sizes available in library and the mismatch between rise- and fall-delay of buffers

5. EXPERIMENTAL RESULTS

5.1 Experimental Setting

We carried out experiments on a set of sequential circuits taken from the ISCAS and ITC benchmarks, as well as on some circuits extracted from open cores [22]. The second and third columns of Table 1 report the number of combinational gates and the number of pulsed-latches of each circuit, after it is synthesized with SIS [23]. A gate library used for technology mapping during the synthesis was constructed for 22 gates, which were based on 45-nm commercial technology. The fourth column shows the initial clock period, where all latches are driven by a single pulse width of 132 ps, which is minimum value available in the technology. The fifth column reports the clock period after (ideal) clock skew scheduling (T_{css}), followed by fixing hold violations; thus, it serves as a lower bound.

We designed five different pulse generators with pulse width of 132-, 192-, 252-, 312-, and 372-ps. Each pulse generator consists of an inverter, a delay cell, and an AND gate; it was designed to drive up to 10 latches with a slew constraint of 45 ps, which was found to be an upper bound for safe latching of data.

5.2 Clock Period

The heuristic algorithm presented in Section 4 was implemented in SIS. The clock period obtained by the heuristic (normalized to T_{css}) is reported in the sixth column of Table 1; it is very close to T_{css} in all examples with average of 1.03×. This was achieved with moderate increase of latches, 16% on average, as reported in the seventh column. The corresponding figures from the conventional retiming [15] are shown in the last two columns as a reference of comparison; the clock period is 1.11× of T_{css} with 27% increase of latches on average. In retiming, a gate delay is typically considered a constant. Retiming move, however, causes the change in the delay of some gates due to the change in their load capacitance and input transition time. This is taken into account in our implementation of retiming by performing an incremental timing analysis; any retiming move that increases, rather than decreases, the clock period is rejected. This is why the number of extra latches introduced by retiming, as reported in the last column, is not significantly large (considering the typical numbers, for example [15]); it also explains the inability of retiming to reduce the clock period beyond that reported in the eighth column. On the other hand, the proposed heuristic approach, which combines retiming and time borrowing, can further reduce the clock period with less use of extra latches due to the flexibility offered by time borrowing.

The distribution of pulse widths after running the heuristic algorithm is shown in Figure 6, which demonstrates the numerical domination of the narrowest pulse of 132 ps. When

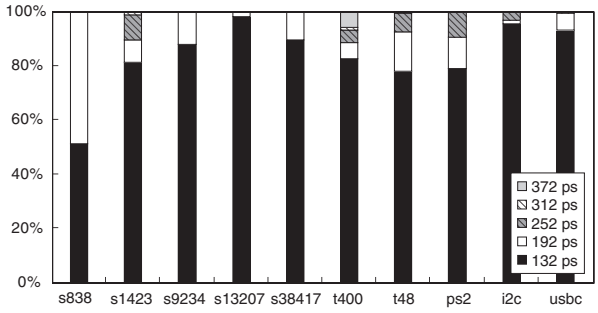

Figure 6: Distribution of pulse widths.

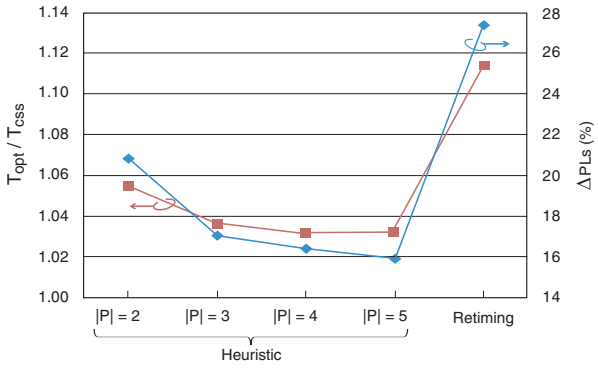

Figure 7: Comparison of T_{opt}/T_{css} and ΔPLs between heuristic with different size of pulse width library ($|\mathcal{P}|$) and retiming.

we restrict the number of available pulse widths to 2, 3, and 4 (starting from 132 ps), the average clock period and the number of extra latches are shown in Figure 7, which also shows the figures from Table 1 ($|\mathcal{P}| = 5$ and retiming) for comparison. As we have more pulse widths available in a library, we have more benefit both in the clock period and the number of extra latches, but with marginal improvement beyond three pulse widths.

The clock period at each step of the heuristic (normalized to T_{css}) is illustrated in Figure 8 for four sample circuits; the initial clock period when we use flip-flops instead of latches is also shown to highlight the benefit of pulsed-latches for a higher performance. It is again clear that T_{opt} can be made close to T_{css} for all examples. The clock period before and after fixing hold time violations remain almost constant in all examples.

6. CONCLUSION

Clock skew has become the unreliable way of optimizing sequential circuits due to increasing within-die process variations with technology scaling. Retiming, on the other hand, has a limitation in practical use due to the large increase of sequential elements. To optimize pulsed-latch-based circuits, we have used both retiming and time borrowing, where the latter is available even in pulsed-latches as long as pulse is wider than the setup time. The ILP approach has been taken to lay a theoretical foundation; the heuristic approach has been proposed for a practical use of optimization. The Monte Carlo simulation of five pulse widths, which we have tried in the experiments, shows the standard deviation of

Table 1: Comparison of the clock period (T_{opt}), normalized to T_{css}, and the number of extra pulsed-latches (ΔPLs) between the proposed heuristic and conventional retiming

Benchmark			T_{ini}	T_{css}	Heuristic		Retiming	
Name	# Gates	# PLs	(ps)	(ps)	T_{opt} (×)	ΔPLs	T_{opt} (×)	ΔPLs
s838	403	32	590	419	1.07	11	1.21	10
s1423	740	74	1733	1510	1.03	13	1.08	14
s9234	1495	135	797	734	1.04	18	1.04	30
s13207	3499	490	981	862	1.01	6	1.07	6
s38417	12458	1463	899	852	1.02	61	1.06	64
t400	2847	176	1104	918	1.02	39	1.20	113
t48	3278	216	1126	968	1.03	76	1.10	138
ps2	2590	185	891	711	1.06	30	1.16	90
i2c	1312	129	980	814	1.01	7	1.03	14
usbc	2619	402	649	541	1.04	38	1.20	33
Average					1.03	16%	1.11	27%

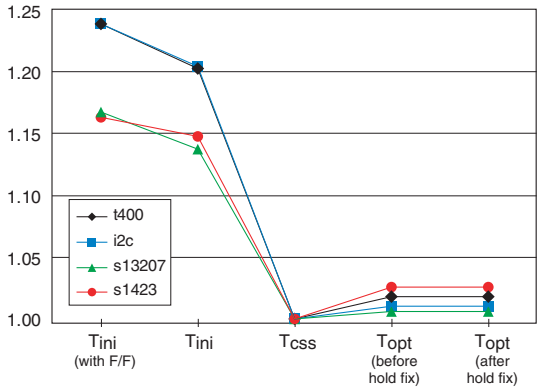

Figure 8: Clock period at each step of the heuristic, normalized to T_{css}.

pulse width ranging from 4.1% to 6.8% of its mean, which suggests that the combined time borrowing and retiming is indeed the reliable way to optimize pulsed-latch circuits.

References

[1] D. Chinnery and K. Keutzer, *Closing the Gap Between ASIC & Custom*, Kluwer Academic Publishers, 2002.

[2] H. Partovi et al., "Flow-through latch and edge-triggered flip-flop hybrid elements," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1996, pp. 138–139.

[3] S. Kozu et al., "A 100 MHz 0.4W RISC processor with 200 MHz multiply-adder, using pulse-register technique," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1996, pp. 140–141.

[4] A. Scherer et al., "An out-of-order three-way superscalar multimedia floating-point unit," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1999, pp. 94–95.

[5] L. Clark et al., "An embedded 32-b microprocessor core for low-power and high-performance applications," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1599–1608, Nov. 2001.

[6] N. Kurd et al., "A multigigahertz clocking scheme for the Pentium 4 microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1647–1653, Nov. 2001.

[7] S. Naffziger et al., "The implementation of the Itanium 2 microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1448–1460, Nov. 2002.

[8] H. Ando et al., "A 1.3-GHz fifth-generation SPARC64 microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 11, pp. 1896–1905, Nov. 2003.

[9] J. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, no. 7, pp. 945–951, July 1990.

[10] K. Carrig, "Chip clocking effect on performance for IBM's SA-27E ASIC technology," *IBM Micronews*, vol. 6, no. 3, pp. 12–16, 2000.

[11] S. Held et al., "Clock scheduling and clocktree construction for high performance ASICs," in *Proc. Int. Conf. on Computer-Aided Design*, Nov. 2003, pp. 232–239.

[12] H. Lee, S. Paik, and Y. Shin, "Pulse width allocation with clock skew scheduling for optimizing pulsed latch-based sequential circuits," in *Proc. Int. Conf. on Computer-Aided Design*, Nov. 2008, pp. 224–229.

[13] C. Chiang and J. Kawa, *Design for Manufacturability and Yield for Nano-Scale CMOS*, Springer, 2007.

[14] C. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming," in *Proc. CalTech Conf. on VLSI*, Mar. 1983, pp. 23–36.

[15] S. Sapatnekar and R. Deokar, "Utilizing the retiming-skew equivalence in a practical algorithm for retiming large circuits," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 10, pp. 1237–1248, Oct. 1996.

[16] S. Sapatnekar, *Timing*, Kluwer Academic Publishers, 2004.

[17] C. Leiserson and J. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1-6, pp. 5–35, June 1991.

[18] D. Singh and S. Brown, "Constrained clock shifting for field programmable gate arrays," in *Proc. Int. Symp. on Field-Programmable Gate Arrays*, Feb. 2002, pp. 121–126.

[19] Y. Kohira and A. Takahashi, "Clock period minimization method of semi-synchronous circuits by delay insertion," in *Proc. Asia-Pacific Conf. on Circuits and Systems*, Dec. 2004, pp. 533–536.

[20] C. Lin and H. Zhou, "Clock skew scheduling with delay padding for prescribed skew domains," in *Proc. Asia South Pacific Design Automation Conf.*, Jan. 2007, pp. 541–546.

[21] N. Shenoy, R. Brayton, and A. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *Proc. Int. Conf. on Computer-Aided Design*, Nov. 1993, pp. 156–161.

[22] http://www.opencores.org/.

[23] E. Sentovich et al. May 1992, Tech. Rep. UCB/ERL M92/41.