Pulse Width Allocation and Clock Skew Scheduling: Optimizing Sequential Circuits Based on Pulsed Latches

Hyein Lee, Seungwhun Paik, *Student Member, IEEE,* and Youngsoo Shin, *Senior Member, IEEE*

***Abstract*—Pulsed latches, latches driven by a brief clock pulse, offer the same convenience of timing verification and optimization as flip-flop-based circuits, while retaining the advantages of latches over flip-flops. But a pulsed latch that uses a single pulse width has a lower bound on its clock period, limiting its capacity to deal with higher frequencies or operate at lower V_{dd}. The limitation still exists even when clock skew scheduling is employed, since the amount of skew that can be assigned and realized is practically limited due to process variation. For the first time, we formulate the problem of allocating pulse widths, out of a small discrete number of predefined widths, and scheduling clock skews, within a predefined upper bound on skew, for optimizing pulsed latch-based sequential circuits. We then present an algorithm called *PWCS_Optimize* (pulse width allocation and clock skew scheduling, PWCS) to solve the problem. The allocated skews are realized through synthesis of local clock trees between pulse generators and latches, and a global clock tree between a clock source and pulse generators. Experiments with 65-nm technology demonstrate that combining a small number of different pulse widths with clock skews of up to 10% of the clock period yield the minimum achievable clock period for many benchmark circuits. The results have an average figure of merit of 0.86, where 1.0 indicates a minimum clock period, and the average reduction in area by 11%. The design flow including *PWCS_Optimize*, placement and routing, and synthesis of local and global clock trees is presented and assessed with example circuits.**

***Index Terms*—Clock period, clock skew scheduling, clock tree, pulsed latch, sequential circuit.**

I. Introduction

FLIP-FLOPS are memory elements that are commonly used in the design of sequential circuits such as finite-state machine controllers and pipelined circuits. Edge-triggered sequential circuits, which consist of combinational blocks that lie between edge-triggered D flip-flops, are the most common form of sequential circuits in application-specific integrated circuit (ASIC) designs due to the convenience with which their timing can be verified. Each combinational block between flip-flops can be identified and its validity of timing constraints can be verified independently from other blocks, which in turn allows independent timing optimization. Flip-flops, however, impose a greater overhead in terms of delay, clock load, and area than latches, as shown in Table I, which was obtained from a SPICE simulation of 1.2-V, 65-nm technology. These overheads are unavoidable since flip-flops are typically constructed by connecting two level-sensitive latches in a master–slave fashion. In particular, the delay of a flip-flop is one of many reasons why ASICs are slower than custom designs in the same technology node by a factor of six or more [1].

Level-sensitive sequential circuits based on latches, while superior to flip-flop-based ones, nevertheless, make timing verification very difficult, since combinational blocks are not isolated from each other due to the transparent nature of latches. On the other hand, this transparency offers more flexibility: latches allow combinational blocks to have a delay more than a clock period, commonly called time borrowing or cycle stealing; and clock skew can be tolerated if the transparency window, shifted by skew, can still capture the data. For this reason, latches are widely used in high-performance microprocessors.

A. Pulsed Latch-Based Circuits

Pulsed latches are latches driven by a brief clock pulse. They retain the design advantage of latches while offering flip-flop-like timing verification and optimization, since they behave like flip-flops due to a short period of transparency.[1] Several types of pulsed latches have been proposed, mostly for high-performance microprocessor designs [2]–[8]. For instance, pulsed latches are used for timing-critical paths while flip-flops are used for the paths that are not critical to timing [8]. The application of pulsed latches to ASICs has been reported [9] recently; the substitution of pulsed latches for some flip-flops can yield a 20% reduction in total dynamic power consumption.

Manuscript received July 28, 2008; revised March 11, 2009 and August 12, 2009. Current version published February 24, 2010. This work was supported by the Korea Science and Engineering Foundation Grant, funded by the Ministry of Education, Science and Technology (MEST), No. R01-2007-000-20891-0. A preliminary version of this paper was presented at the International Conference on Computer-Aided Design, San Jose, CA, November 10–13, 2008. This paper was recommended by Associate Editor L. Scheffer.

H. Lee is with Samsung Electronics, Yongin, Gyeonggi-Do 449-711, Korea (e-mail: hyein23.lee@samsung.com).

S. Paik and Y. Shin are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: swpaik@dtlab.kaist.ac.kr; youngsoo@ee.kaist.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2041845

[1]Ideally, pulsed latches become edge-triggered devices for a pulse of zero width. However, in practice the pulse width has to be large enough for latches to capture the data safely.

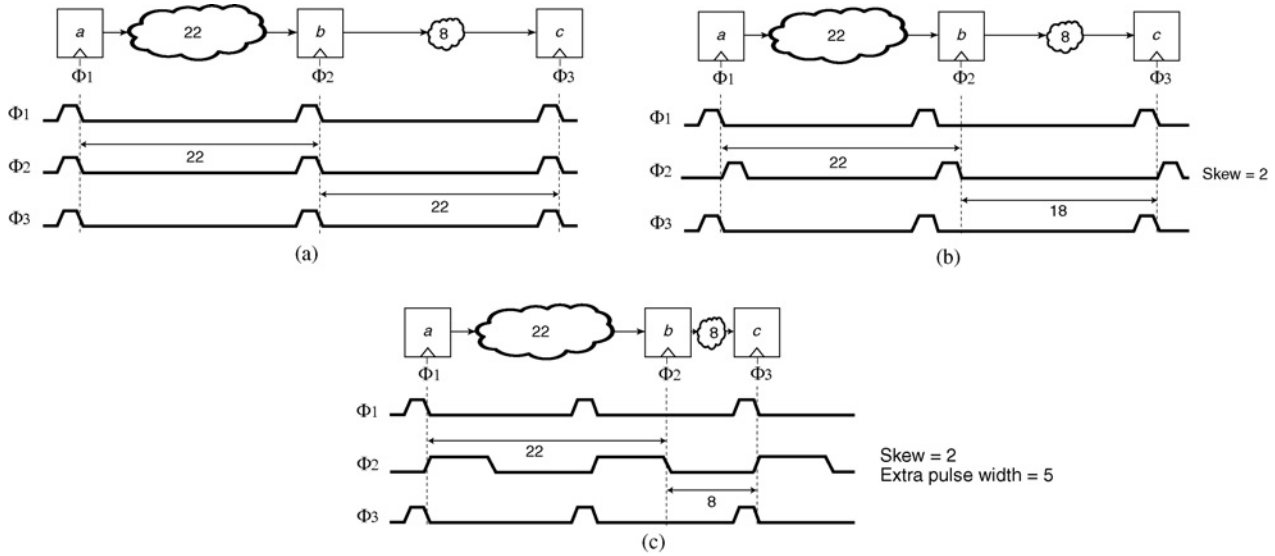


Fig. 1. Motivational example of a pulsed latch-based circuit. (a) Single pulse width without clock skew. (b) Single pulse width with a skew of up to two time units. (c) Multiple pulse widths with a skew of up to two time units.

TABLE I
COMPARISON OF AN EDGE-TRIGGERED D FLIP-FLOP AND A LEVEL-SENSITIVE D LATCH IN 65-NM TECHNOLOGY

	F/F	Latch
Setup time (ps)	72	47
Clock-to-Q delay (ps)	238	190
Clock load (fF)	3.4	2.2
Area (μm^2)	8.0	4.8

There are two approaches to generating a clock pulse: either it can be internally generated by an individual pulsed latch [2], [4] or external pulse generators [3], [7], which receive a normal clock and generate a clock pulse, can be used to deliver the clock pulse to local pulsed latches that are physically close to the pulse generators.

B. Motivation

Pulsed latch-based circuits using a single pulse width, which is the conventional approach, cannot take advantage of time borrowing due to their short period of transparency. This can be alleviated by employing sequential optimization techniques such as retiming [10] or clock skew scheduling [11]. However, the use of retiming often causes a large increase in the number of latches [12] thus limiting its practical use; and it can also have an impact on the verification methodology [13]. Conventional clock skew scheduling [15] assigns an arbitrary amount of skew to each latch to balance the delay between the combinational blocks. But this approach has become impractical because within-die (WID) variations, which grow with technology scaling [16], affect the extra buffers and wires inserted to implement large skews in randomly different amount, thereby causing uncertainties in the skews [13], [17]. It has been shown that the maximum difference in clock arrival times that can be practically realized are less than 10% of the clock period in 0.18-μm technology [17], or 10% to 16% in 0.18-μm and 0.13-μm technologies [18]. This is also true in a clock grid [19], where only a very small amount of skew can be realized.

In this paper, we exploit multiple pulse widths together with clock skews to minimize the clock period of pulsed latch-based circuits, where the pulse width is defined by the pulse generator which drives the latch and the skew is constrained by an upper bound defined as a fraction of the clock period. The rationale behind the approach is that the clock pulse delivered by pulse generators is less susceptible to WID process variations. The impact of systematic components of WID variations is more severe with clock buffers because they can be located far from each other whereas the delay cells of a pulse generator are located within the pulse generator itself. Moreover, pulse generators can be further tailored for robust operation under *PVT* variations [20].

Fig. 1 explains the motivation for the proposed approach. We consider two combinational blocks: 1) between latches a and b with a maximum delay of 22 time units; and 2) between latches b and c with a maximum delay of 8. Fig. 1(a) shows three pulsed latches driven by a single pulse width and without any clock skew; the clock period has to be at least 22, if the setup time and clock-to-Q delay are assumed to be 0. If the arrival time of the pulse at b is intentionally delayed by 2, which is assumed to be the maximum skew allowed, the clock period can be reduced to 20, as shown in Fig. 1(b). Note that the combinational block between b and c still has a slack of 10, which illustrates the limitation of clock skew scheduling when maximum skew is limited. In Fig. 1(c), the same amount of skew is applied to b, but using a different pulse, with a width which is 5 greater than that of the pulse applied to a and c; this allows the combinational block between a and b to borrow time from the block between b and c, which eventually yields a smaller clock period of 15. This is the minimum clock period for this particular example.

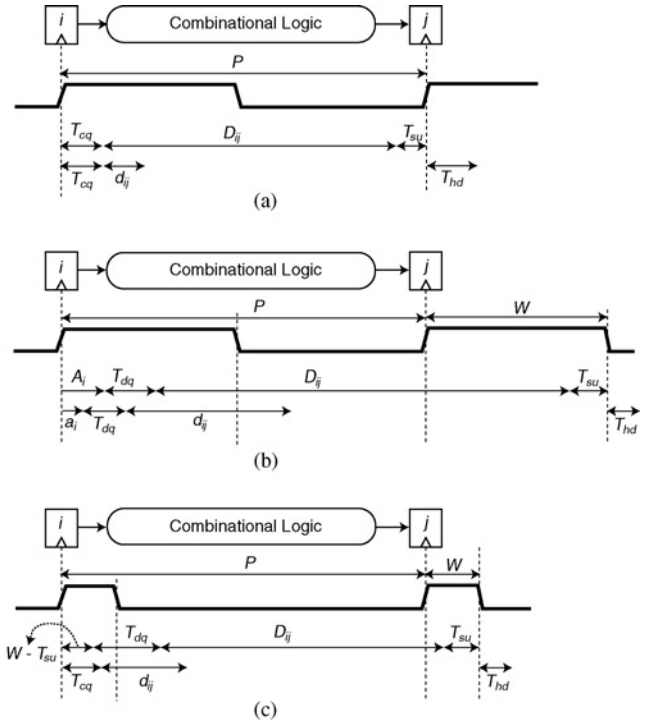


Fig. 2. Setup and hold time constraints. (a) Flip-flop-based circuits. (b) Latch-based circuits. (c) Pulsed latch-based circuits.

C. Contributions

The main contributions of this paper are as follows.

1) The definition of pulse width allocation and clock skew scheduling (PWCS) (Section III-C), to minimize the clock period of pulsed latch-based circuits, and an algorithm called *PWCS_Optimize* (Section III-D) that solves the problem.
2) Latch clustering and clock tree synthesis, which synthesizes local clock trees between pulse generators and latches, and a global clock tree between a clock source and pulse generators, to realize the allocated skews (Section III-E).
3) Experiments with commercial 65 nm technology (Section IV), which demonstrates that a small number of different pulse widths of up to 5, combined with clock skews of up to 10% of the clock period, yield an average figure of merit of 0.86, where 1.0 indicates a minimum clock period, for several benchmark circuits, while reducing the area requirement by 11% on average.

The remainder of this paper is organized as follows. In Section II, we present a brief overview of the timing constraints of sequential circuits based on flip-flops, latches, and pulsed latches. The problem of pulse width allocation and clock skew scheduling is addressed in Section III, together with the *PWCS_Optimize* algorithm and an algorithm for latch clustering and clock tree synthesis. Experimental results are presented in Section IV, where we discuss the effectiveness of *PWCS_Optimize* in reducing the clock period, and its impact on physical design and power consumption; conclusions are drawn in Section V.

II. Timing Constraints of Sequential Circuits

In this section, we review the timing constraints of sequential circuits, setup and hole time constraints, based on flip-flops, latches, and pulsed latches, with the help of Fig. 2. The setup time is denoted by T_{su}, the hold time by T_{hd}, the clock-to-Q delay by T_{cq}, the data-to-Q delay by T_{dq}, and the clock period by P. We do not distinguish between propagation and contamination delays, i.e., maximum and minimum delay, of T_{cq} and T_{dq} for simplicity of presentation. The timing parameters of all sequencing elements in each type of circuit are assumed to be equal. The maximum delay of the combinational block between sequencing elements i and j is denoted by D_{ij}, and the minimum delay by d_{ij}.

A. Flip-Flop-Based Circuits

In positive edge-triggered sequential circuits, data are launched from a flip-flop i at the rising edge of the clock and the latest result of computation from the combinational block has to arrive at flip-flop j earlier than the setup time before the next rising edge of the clock [see Fig. 2(a)]. This constitutes the setup time constraint

$$T_{cq} + D_{ij} \leq P - T_{su}. \tag{1}$$

The earliest data from the combinational block have to arrive at j no earlier than its hold time after the rising edge of the clock, so that j can hold its current data in a stable state. This constitutes the hold time constraint

$$T_{cq} + d_{ij} \geq T_{hd}. \tag{2}$$

B. Latch-Based Circuits

In a single-phase positive level-sensitive sequential circuit, data can arrive at any time when the clock is high, unless it is later than the setup time before the falling edge of the clock. Let A_i be the latest data arrival time at latch i, which can be computed iteratively from the data arrival times at all the latches that are connected to i through combinational blocks [14]

$$A_i = \max_{\forall_{k \rightsquigarrow i}} \left[\max\left(T_{cq}, A_k + T_{dq}\right) + D_{ki}\right] \tag{3}$$

where T_{cq} corresponds to data arriving at k before the rising edge and $A_k + T_{dq}$ after the rising edge. The setup time constraint between latches i and j can then be described [21] by

$$\max\left(T_{cq}, A_i + T_{dq}\right) + D_{ij} \leq P + W - T_{su} \tag{4}$$

where W is the period of the clock being high [see Fig. 2(b)]. Note that a time borrowing of up to $W - T_{su}$ is implicitly allowed in this constraint.

Similarly, if the earliest data arrival time at i is denoted by a_i, which can be computed by

$$a_i = \min_{\forall_{k \rightsquigarrow i}} \left[\max\left(T_{cq}, a_k + T_{dq}\right) + d_{ki}\right] \tag{5}$$

then the hold time constraint can be described by

$$\max\left(T_{cq}, a_i + T_{dq}\right) + d_{ij} \geq W + T_{hd}. \tag{6}$$

C. Pulsed Latch-Based Circuits

The reason why timing constraints for latch-based circuits [(4) and (6)] are complicated is because of time borrowing, i.e., some combinational blocks between latch pairs use more than a clock period, which has to be compensated for by some other combinational blocks using less than a clock period. The timing constraints would become similar to those of flip-flop-based circuits [(1) and (2)] if time borrowing were not allowed, i.e., if all the combinational blocks were forced to use no more than a clock period, but this would take away the benefit of using latches in high-performance designs.

Since the amount of the time borrowing is determined by W, and pulsed latches are latches driven by a clock with a very short W, we can safely choose to use flip-flop-like timing constraints by preventing time borrowing for the convenience of setting up timing constraints. Let us assume that T_{su} is smaller than the pulse width W, as shown in Fig. 2(c), which is usually true in practice as we will see in Section IV. Each combinational block is allocated a clock period between $W - T_{su}$, which is the latest time at which data can depart from i, and $P + W - T_{su}$, which is the latest time at which data can arrive to j [see Fig. 2(c)]. This yields the setup time constraint

$$(W - T_{su}) + T_{dq} + D_{ij} \leq P + (W - T_{su}). \tag{7}$$

Note that this constraint is similar to (1) if we remove $W - T_{su}$ from both sides of the inequality. The earliest time at which data can depart is the rising edge of the clock, which leads to the hold time constraint

$$T_{cq} + d_{ij} \geq W + T_{hd}. \tag{8}$$

III. Pulse Width Allocation and Clock Skew Scheduling

A. Overview

Pulsed latch-based circuits based on the timing constraints (7) and (8), which assume that all latches have the same pulse width W, have no advantages over flip-flop-based circuits, except that the sequencing elements have superior design parameters. However, if we allow a small variety of different pulse widths and assign an intentional clock skew to each pulsed latch, which is our model of pulsed latch-based circuits, there is a potential to reduce the clock period, as we suggested in our discussion of Fig. 1.

This approach to optimizing pulsed latch-based sequential circuits is illustrated in Fig. 3. We receive a gate-level netlist of a circuit which has been synthesized with initial timing constraints, which include the clock period as an input. A list of available pulse widths is defined by a library of pulse generators; clock skews are restricted to within a specified upper bound.

Once each pulsed latch has been assigned a pulse width and a clock skew, the latches with the same pulse width are placed in groups containing the maximum number of latches that can be driven by a single pulse generator, and each group is assigned to a specific pulse generator. The nets between each group of latches and the pulse generator that drives them are assigned a higher net weight, so that they are placed close during the subsequent automatic placement.[2] After automatic routing is performed, the skews assigned to latches are realized through the synthesis of local clock trees between pulse generators and latches, and the synthesis of a global clock tree between a clock source and the pulse generators. Note that part of the skew is realized by means of a local clock tree, while the global clock tree is responsible for the remaining skew. The details of this design flow will be explained in Section III-E.

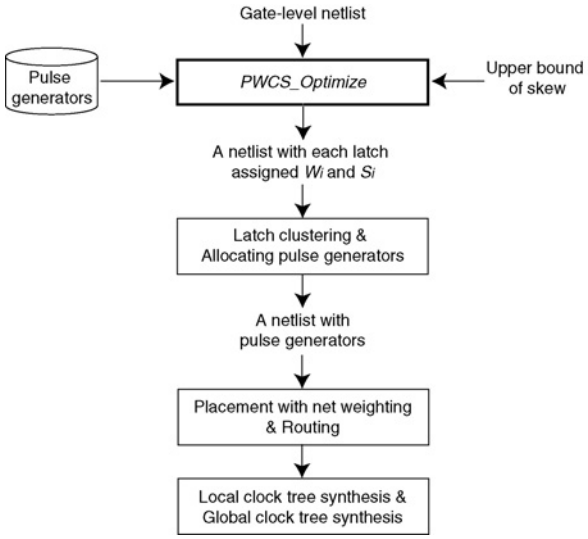


Fig. 3. Overall flow of optimizing pulsed latch-based sequential circuits.

B. Timing Constraints for Multiple Pulse Widths and Clock Skew

If an intentional clock skew, denoted by S_i, can be assigned to each latch i, the original setup time and hold time constraints, (7) and (8), respectively, become

$$S_i + T_{dq} + D_{ij} \leq P + S_j \tag{9}$$

$$S_i + T_{cq} + d_{ij} \geq S_j + W + T_{hd}. \tag{10}$$

For a given clock period P, (9) and (10) can be verified in polynomial time [22] to check whether there is a feasible set of S_is. The minimum P, under the assumption that S_i takes an arbitrary value, can be derived by verifying (9) and (10) $\log_2 N$ times [22], where N is the number of potential clock periods that can be tried. If the values of S_is are restricted, e.g., to 10% of the clock period [17], the smallest P that can be achieved may be far larger than the minimum clock period as we mentioned in discussing Fig. 1(b).

To overcome this limitation, we further assume that we can allocate a different pulse width $W_i \in \mathcal{W}$, where $\mathcal{W}$ is a list of

[2]This has the negative effect of increasing signal wires, as reported in Section IV. An alternative approach would be to perform placement without pulse generators, and then performing in-placement allocation of pulse generators. In general, this would increase the number of pulse generators, since the connection between a pulse generator and its latches should be localized to avoid distortion of the pulse shape during transmission over a long distance.

available pulse widths, to each latch i, which yields the refined constraints

$$S_i + W_i + T_{dq} + D_{ij} \leq P + S_j + W_j \tag{11}$$

$$S_i + T_{cq} + d_{ij} \geq S_j + W_j + T_{hd}. \tag{12}$$

Note that the introduction of W_i and W_j in (11) has the effect of reinforcing time borrowing, since $W_i < W_j$ implies that the combinational block between i and j can use more than a clock period [see Fig. 1(c)].

C. Problem Formulation

With setup time and hold time constraints specified by (11) and (12), respectively, we now state the problem of minimizing the clock period by allocating W_i and S_i.

Problem 1: Given a netlist of a sequential circuit with timing constraints consisting of arrival times at primary inputs and required arrival times at primary outputs, a list of distinct pulse widths $\mathcal{W}$, and a maximum allowable clock skew Δ, the *PWCS optimization problem* is to allocate a pulse width $W_i \in \mathcal{W}$ and assign a clock skew $S_i \leq \Delta$ to each pulsed latch i. The objective of the *PWCS optimization problem* is to minimize the clock period while satisfying the setup time and hold time constraints as described by (11) and (12).

Problem 1 can be solved by verifying iteratively, through binary search, whether we can find a feasible set of (W_i, S_i) for one particular clock period. This requires $\log_2 (P_{\max} - P_{\min}) / \epsilon$ tries, where $P_{\max}$ and $P_{\min}$, respectively, are the upper and lower bounds of the clock period that can be tried and ϵ is an increment of the clock period, e.g., 1 ps. The upper bound $P_{\max}$ can be derived from (11) by setting all pulse widths equal and all skews to 0 [22]

$$P_{\max} = \max_{\forall_{i \rightsquigarrow j}} \left(T_{dq} + D_{ij} \right). \tag{13}$$

The longest path from any latch to itself can serve as a lower bound. The shortest path from any latch to some other latch also serves as a lower bound [22]. The maximum value of these two is $P_{\min}$

$$P_{\min} = \max \left[\min_{\substack{\forall_{i \rightsquigarrow j}, \\ i \neq j}} \left(T_{dq} + D_{ij} \right), \max_{\forall_{i \rightsquigarrow i}} \left(T_{dq} + D_{ii} \right) \right]. \tag{14}$$

Each iteration of the binary search is defined as a PWCS problem, i.e., a problem that verifies whether a feasible set of (W_i, S_i) exists for one particular clock period.

Problem 2: Given a netlist with timing constraints, a list of distinct pulse widths $\mathcal{W}$, and a maximum allowable clock skew Δ, the *PWCS problem* is to find $W_i \in \mathcal{W}$ and $S_i \leq \Delta$ such that a specified clock period P can be satisfied under (11) and (12).

D. Algorithm

The algorithm *PWCS_Optimize*, which we use to solve Problem 1, is shown in Fig. 4. It iteratively checks the median clock period (L3) between the current maximum P_u and minimum P_l, by calling the function *PWCS* (L4). If the period turns out to be feasible, it serves as a new maximum (L4) for the next iteration, otherwise it serves as a new minimum (L5).

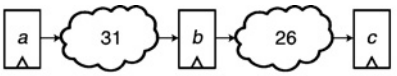
```
Algorithm PWCS_Optimize (P_max, P_min, ε)
begin
L1      P_u := P_max,   P_l := P_min
L2      while (P_u − P_l) > ε do
L3          P := (P_u + P_l) /2
L4          if PWCS (P) = success then  P_u := P
L5          else P_l := P
            end if
        end do
end

Function PWCS (P)
begin
L6      Sort a list of pulse widths W in order of increasing width
L7      W_i := min_{x∈W} x,  S_i := 0,   ∀_i
L8      C(i, j): W_j + S_j ≥ W_i + S_i + (T_dq + D_ij − P) , ∀_{i⇝j}
L9      while ∃_{i,j}, C(i, j) is not satisfied do
L10         r_h := W_i + S_i + (T_dq + D_ij − P)
L11         Increment W_j until W_j + Δ ≥ r_h
L12         if such W_j ∈ W does not exist then  return fail
            end if
L13         S_j := max (0, r_h − W_j)
        end do
L14     return success
end
```

Fig. 4. Pseudocode of the PWCS optimization algorithm.

The function *PWCS*, which solves Problem 2, is also shown in Fig. 4. In the implementation of this function, we consider the setup time constraint (11) alone, and ignore the hold time constraint (12); this approach is common to many methods of clock skew scheduling [22]–[24]. As is shown in [22], considering only the setup time constraint always yields a smaller clock period than considering both the setup and hold time constraints, which is the motivation of our choice of implementation. Once the clock period has been determined, the extra buffers are introduced into logic paths that violate hold time constraints [25], even though that may increase the determined clock period, due to the finite number of buffer sizes available in the library and the mismatch between rise and fall-delay of buffers, which is discussed in Section IV-A.

The function *PWCS* works as follows. A list of available pulse widths ($\mathcal{W}$) is arranged in order of increasing width (L6). All latches are initialized to have minimum pulse width and zero clock skew (L7). For each pair of latches i and j, the setup time constraint (11) is constructed (L8); this is denoted by $C(i, j)$ in the algorithm. We then iterate through the loop for each unsatisfied constraint $C(i, j)$ (L9). The right-hand side of the inequality $C(i, j)$, denoted by r_h, is considered to be fixed (L10); and we regard the parameters W_j and S_j of the data-capturing latch as variables. This is essentially an iterative relaxation-based version of the Bellman-Ford algorithm, which has been shown to be optimal [26] in the sense that P is feasible if and only if *PWCS* returns successfully. Therefore, the number of iterations of L9 has the bound $O(nm)$, where n is the number of latches and m is the number of pairs of launching and capturing latches. *PWCS_Optimize* calls *PWCS* $\log_2 (P_{\max} - P_{\min}) / \epsilon$ times, thus the bound on



(a)

Iter.	(W_a, S_a)	(W_b, S_b)	(W_c, S_c)	Constraint
0th	(5, 0)	(5, 0)	(5, 0)	
1st	(5, 0)	(5, 0)	(5, 1)	$W_c + S_c \geq W_b + S_b + 1 = 6$
2nd	(5, 0)	(10, 1)	(5, 1)	$W_b + S_b \geq W_a + S_a + 6 = 11$
3rd	(5, 0)	(10, 1)	(10, 2)	$W_c + S_c \geq W_b + S_b + 1 = 12$

(b)

Fig. 5. (a) Example sequential circuit. (b) Iterations of *PWCS*.

PWCS_Optimize is $O(nm \log_2 (P_{\max} - P_{\min})/\epsilon)$. In practice, however, $\log_2 (P_{\max} - P_{\min})/\epsilon$ is independent of the problem size and thus can be considered as a constant; since it was smaller than 16 for all benchmark circuits tested, we can assume that *PWCS_Optimize* also runs in $O(nm)$.

We now need to determine values of W_j and S_j with a sum that is not smaller than r_h (L11). We want to make this sum as small as possible as long as it is larger than or equal to r_h, so that W_j and S_j become less restrictive when j is located on the right-hand side (r_h) in later iterations. Since W_j takes one of the discrete number of values in $\mathcal{W}$, while S_j takes any value between 0 and Δ, the number of combinations of (W_j, S_j) that satisfy $C(i, j)$ is discrete. Combinations with a smaller W_j are preferred, since the overhead of the pulse generator increases with increasing pulse width, which is made clear in Section IV. Therefore, we assume the maximum skew (Δ) and try to find the smallest pulse width that satisfies the constraint (L11). If such a pulse width does not exist, then $C(i, j)$ cannot be satisfied and the function *PWCS* terminates in failure (L12). Otherwise, S_j is set to its smallest value (L13) and the loop is repeated. Observe that some $C(i, j)$s that are satisfied at the end of an early iteration can violate criteria in later iterations, and therefore need to be determined again.

Example 1: Consider the example circuit shown in Fig. 5(a). The maximum delays of each of two combinational blocks are denoted by 31 and 26 (D_{ij}). We want to perform *PWCS* for a clock period of 26 time units. Let us assume that T_{dq} is 1 time unit, that the maximum allowable skew (Δ) is 3 time units, and that the pulse widths can be selected from $\mathcal{W} = \{5, 10\}$. With an initial pulse width of 5 time units and skew of 0 for all three latches, it can be readily verified that both pairs of latches $a \rightsquigarrow b$ and $b \rightsquigarrow c$ violate the setup time constraint (11). Suppose that we select $b \rightsquigarrow c$ for the first iteration (L9). As shown in Fig. 5(b), r_h is 6 time units (L10), which yields $W_c = 5$ (L11: note that $W_j + \Delta \geq 6$ returns $W_j = 5$ for $\Delta = 3$) and $S_c = 1$ (L13). Further suppose that we select $a \rightsquigarrow b$ for the second iteration, which yields $W_b = 10$ and $S_b = 1$; this causes $b \rightsquigarrow c$ to violate the constraint again, which we select and fix in the third iteration, as shown in Fig. 5(b), and *PWCS* finally returns successfully. □

E. Latch Clustering and Clock Tree Synthesis

Once the pulse width and skew have been determined for each pulsed latch by the *PWCS_Optimize* algorithm, the latches with the same pulse width have to be grouped so that they can be driven by a single pulse generator (see the second step of Fig. 3). There is an upper bound on the number of latches that can be driven by one pulse generator, which is 10 in the experiments reported in Section IV. Thus, latches with the same pulse width, if their numbers exceed this bound, are evenly distributed across several pulse generators. While we group the latches, we take their skews into account. All the latches with same pulse width are initially arranged in order of increasing skew. When we distribute them across several pulse generators, we keep this order, so that latches of similar skews are driven by the same pulse generator. For instance, if four latches have the same pulse width and skews of 1, 2, 3, and 4, and one pulse generator can drive two latches, then the latches of skew 1 and 2 are driven by one pulse generator while latches of skew 3 and 4 by the other.

Algorithm *Clock_Tree_Synthesis* $(\phi_1, \phi_2, \ldots, \phi_n)$
begin
L1 **for** $i = 1, 2, \ldots, n$ **do**
L2 $\rho_i := \min_{j \in \phi_i} S_j$
L3 $S_j := S_j - \rho_i, \ \forall_{j \in \phi_i}$
Local clock tree synthesis of ϕ_i
end do
L5 *Adjust_PG_Skew* ()
L6 Global clock tree synthesis of $\rho_1, \rho_2, \ldots, \rho_n$
end

Function *Adjust_PG_Skew* ()
begin
L7 **for** $i = 1, 2, \ldots, n$ **do**
L8 $D(\phi_i) := \min_{j \in \phi_i}$ delay from PG to j
end do
L9 $\rho_i := \rho_i + \max_j D(\phi_j) - D(\phi_i), \quad i = 1, 2, \ldots, n$
end

Fig. 6. Pseudocode of clock tree synthesis algorithm.

Once we have grouped the latches, we impose a higher net weight, which is 2 in the experiments, on the nets connection the latches and pulse generators, so that they have a higher possibility of being physically close during automatic placement, which is the third step shown in Fig. 3. This is followed by synthesis of local and global clock trees.

Let groups of latches be denoted by $\phi_1, \phi_2, \ldots, \phi_n$, where n is the total number of groups, and thus also the total number of pulse generators. The algorithm *Clock_Tree_Synthesis* shown in Fig. 6 synthesizes local and global clock trees. Within each group of latches (L1), we first find the latch with minimum skew, denoted by ρ_i (L2). We then subtract ρ_i from the skews of all the latches in the group (L3); the value of ρ_i, after adjustment via *Adjust_PG_Skew*, ends up as a skew applied to the pulse generator in the global clock tree. This leaves a minimum amount of skew for each latch in the local clock tree. Unnecessarily large skews in the local clock tree between pulse generator and latches, which may need long wires and many buffers, could cause distortion of the pulse shape. The new skews obtained from L3 are then submitted to a conventional clock tree synthesis tool (L4) [27].

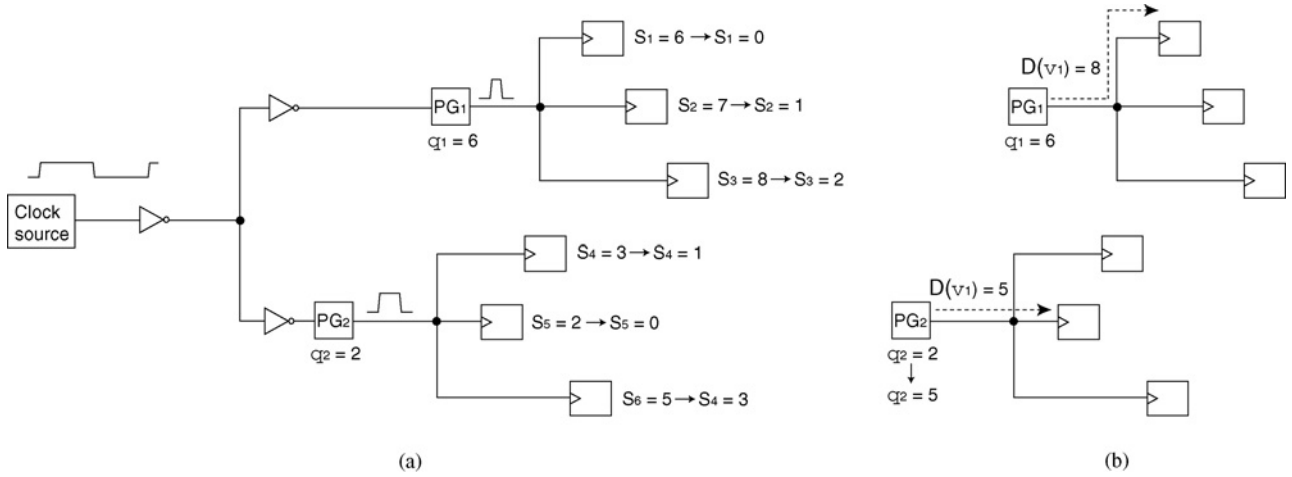


Fig. 7. Example of clock tree synthesis: (a) adjusting skews of latches $S_1, S_2, \ldots, S_6$ for synthesis of two local clock trees, and (b) adjusting the skews of pulse generators ρ_1 and ρ_2 for synthesis of the global clock tree.

Since each local clock tree is synthesized independently (L4) and the adjusted skews are relative and only valid within a particular local clock tree, the same skews in different local clock trees may correspond to different global skews. To compensate for this, the skews of the pulse generators are adjusted by the function *Adjust_PG_Skew* (L5). In each group of latches (L7), we find the delays from the pulse generator to each latch and pick the smallest one (L8), denoted by $D(\phi_i)$, which is the delay to the latch which previously had the minimum skew, and now has zero skew. The difference between the maximum delay for all these zero-skew latches from all groups ($\max_j D(\phi_j)$) and $D(\phi_i)$ is added to the skew of pulse generator ρ_i (L9). The adjusted skews of the pulse generators are then submitted to synthesis of the global clock tree between a clock source and pulse generators (L6).

Example 2: Fig. 7(a) shows latches grouped in two sets: $\phi_1 = \{1, 2, 3\}$ and $\phi_2 = \{4, 5, 6\}$. S_1 is the minimum skew in the first group, thus $\rho_1 = S_1 = 6$ (L2) is subtracted from S_1, S_2, and S_3 (L3) as shown in Fig. 7(a); similarly, $\rho_2 = S_5 = 2$ is subtracted from all skews in the second group. After synthesis of the two local clock trees, the delay between PG_1 and latch 1 is the smallest in ϕ_1, and ϕ_2 has the smallest delay between PG_2 and latch 5, since the skews of both latches are 0. Suppose that these delays are 8 and 5, respectively, as shown in Fig. 7(b). Since the skews of both latches are zero, while the delays from the pulse generators are different, the skew of PG_2 (ρ_2) is adjusted to $2 + (8 - 5) = 5$, as shown in Fig. 7(b). □

IV. Experimental Results

We carried out experiments on a set of sequential circuits taken from the ISCAS and the ITC benchmarks. We also included circuits extracted from several open cores [28] including a communication controller (`i2c`), a direct memory access (`dma`) controller, a keyboard interface unit (`ps2`), two microprocessor units (`t400` and `t48`), a controller area network (`can`) protocol controller, a universal serial bus (USB) controller (`usbc`), and a USB core (`usbf`). The first three columns of Table III give the name, the number of combinational gates, and the number of sequencing elements for each circuit.

TABLE II
Pulse Generators Used in the Experiments

Name	Pulse width (ps)	Area (μm^2)
PG1	156	5.12
PG2	261	5.44
PG3	348	5.76
PG4	447	6.08
PG5	556	6.40

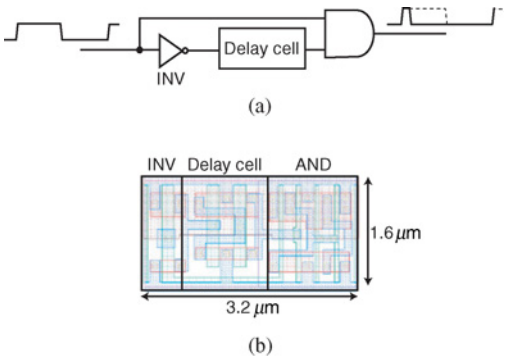


Fig. 8. (a) Pulse generator. (b) Layout of PG1.

Each circuit was synthesized with SIS [29]. The gate library used for technology mapping during the synthesis was constructed for 114 gates, all based on 65-nm commercial technology. The synthesized gate-level netlist was then submitted to the *PWCS_Optimize* algorithm, which we implemented in SIS. This was followed by automatic placement and routing [27]. The latch clustering and clock tree synthesis described in Section III-E were performed via Tcl script on a commercial physical design tool [27].

A set of five pulse generators were constructed, as summarized in Table II. The design considerations relating to the interval of pulse widths will be discussed in Section IV-B. Each pulse generator consists of an inverter, a delay cell, and

TABLE III
COMPARISON OF CLOCK PERIODS FOR PULSED LATCH-BASED CIRCUITS OPTIMIZED WITH *PWCS_Optimize* AND CLOCK PERIODS FOR FLIP-FLOP-BASED CIRCUITS OPTIMIZED BY CLOCK SKEW SCHEDULING

Benchmark			Pulsed Latch-Based Design				Flip-Flop-Based Design			
Name	# Gates	# Seq. Elements	P_{ini} (ps)	P_{opt} (ps)	P_{pwcs} (ps)	η	P_{ini} (ps)	P_{opt} (ps)	P_{css} (ps)	η'
s1423	544	74	3080	2537	2605	0.87	3242	2650	2978	0.45
s9234	1171	135	1455	1350	1350	1.00	1592	1488	1488	1.00
s13207	2998	490	1844	1367	1553	0.61	1993	1453	1848	0.27
s15850	3805	515	2420	1801	1952	0.75	2396	1899	2207	0.38
s38584	11988	1424	2153	1873	1873	1.00	2285	2031	2082	0.80
b04	623	66	1403	1055	1105	0.86	1540	1138	1426	0.28
b11	438	30	1384	1239	1239	1.00	1520	1397	1400	0.97
i2c	1057	129	1651	1501	1501	1.00	1781	1616	1619	0.98
dma	1985	198	1433	1202	1202	1.00	1549	1307	1432	0.48
ps2	1992	185	1704	1164	1297	0.75	1870	1286	1742	0.22
t400	2149	176	1983	1551	1656	0.76	2133	1771	1956	0.49
t48	2533	216	2098	1750	1818	0.80	2223	1895	2033	0.58
can	5150	877	2023	1333	1625	0.58	2184	1437	2041	0.19
usbc	2172	402	1318	976	1028	0.85	1395	1082	1287	0.34
usbf	14770	1733	2526	2213	2213	1.00	2688	2295	2458	0.59
Average						0.86				0.53

an AND gate, as shown in Fig. 8(a); the inverter and delay cell are implemented in high-V_t while the AND gate is in regular-V_t. The pulse width is controlled by the delay cell. Each pulse generator was designed to drive up to 10 latches with a slew constraint of 60 ps, which was found to be the upper bound that would ensure the safe latching of data. The layout of one pulse generator is shown in Fig. 8(b).

A. *Effectiveness of PWCS_Optimize in Reducing the Clock Period*

The results of *PWCS_Optimize* are shown in columns 4–7 of Table III. The initial clock period after logic synthesis, where we assumed the same pulse width for all the latches, is denoted by P_{ini} and is shown in column 4. P_{opt} in column 5 is the optimum clock period, which was obtained after clock skew scheduling on the initial netlist, with the assumption that an arbitrary amount of skew can be assigned. The clock period obtained by *PWCS_Optimize*, followed by a process of fixing hold time violations, is denoted by P_{pwcs} and is shown in column 6. The maximum allowable skew (Δ) was assumed to be 10% of P_{opt} [13], [17]. Out of five pulse generators in Table II, only those with pulse widths within 40% of P_{opt} were allowed for each circuit, so that any path that violates the hold time constraint and thus needs to be fixed is constrained within 50% of P_{opt}. In order to assess the effectiveness of *PWCS_Optimize* in reducing a clock period, we introduce a figure of merit (FOM), which reflects the extent to which a clock period can be reduced with respect to the one achieved by P_{opt}

$$\eta = \frac{P_{ini} - P_{pwcs}}{P_{ini} - P_{opt}}. \tag{15}$$

Column 7 shows the FOM of each circuit. Note that $0 \leq \eta \leq 1$; the larger η is, the closer P_{pwcs} is to P_{opt}.

The FOMs before and after fixing hold time violations are shown in columns 2 and 3 of Table IV; column 3 corresponds to column 7 of Table III. Note that the clock period returned by *PWCS_Optimize* may cause hold time violations, since we do not take hold time constraints into account in the function *PWCS*; the totals of hold time constraints and violations are shown in columns 4 and 5. The violations are resolved by adding extra buffers in the affected timing paths [25], and this process was also implemented in SIS; the number of buffers that were inserted are shown in the last column of Table IV. Due to the finite number of buffer sizes available in the library and the mismatch between the rise and fall-delays of the buffers, extra buffers can increase the clock period, and this occurred with six circuits (s1423, s13207, s15850, t400, t48, and can). Three circuits (s13207, ps2, and can) have rather a small η, below 0.8, even before the hold time violations are fixed. This is because the maximum skew value used in these designs to obtain P_{opt} is considerably larger than the maximum extent of clock skew and pulse width that *PWCS_Optimize* can use, which is a skew up to 10% of P_{opt} plus the maximum pulse width used by the designs subtracted by the minimum pulse width (i.e., 156 ps in our experiment), to reduce the clock period. We compared the maximum delay of adjacent combinational blocks (i.e., D_{ij} and D_{jk} for consecutive latches i, j, and k) and found that three of the circuits have many adjacent blocks with large differences in their maximum delays, which is why a large skew is assigned to minimize P_{opt}, unlike the other circuits.

The distribution of pulse generators after running *PWCS_Optimize* is shown in Fig. 9, which demonstrates the numerical domination of PG1, the pulse generator with the narrowest pulse width. If all the pulse generators were restricted to PG1 alone, the average η would be 0.51; thus, even though PG1 dominates in numbers, exploiting the small number of pulse generators with wider pulse widths allow a significant reduction of the clock period, resulting in the average η of 0.86 shown in Table III.

Similar experiments were performed for flip-flop-based circuits. In the initial netlist synthesized with latches, which

TABLE IV
FOMS BEFORE AND AFTER FIXING HOLD TIME VIOLATIONS

Name	FOM (η)		# of	# of	# of
	Before Fix	After Fix	Hold Time Constraints	Hold Time Violations	Extra Buffers
s1423	1.00	0.87	74 096	210	82
s9234	1.00	1.00	296 508	24	26
s13207	0.72	0.61	2 030 274	146	170
s15850	0.94	0.75	4 532 317	340	374
s38584	1.00	1.00	1 757 704	165	177
b04	0.86	0.86	64 794	34	67
b11	1.00	1.00	14 812	1	1
i2c	1.00	1.00	6444	1	1
dma	1.00	1.00	14 482	74	86
ps2	0.75	0.75	148 780	68	80
t400	1.00	0.76	161 162	144	204
t48	1.00	0.80	482 182	70	75
can	0.61	0.58	1 401 940	41	27
usbc	0.85	0.85	8456	127	200
usbf	1.00	1.00	1 451 524	524	268
Average	0.92	0.86			

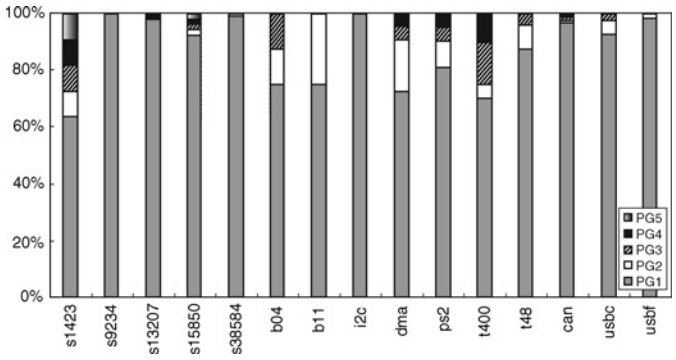


Fig. 9. Distribution of pulse generators after running *PWCS_Optimize*.

corresponds to column 4, we substituted flip-flops for all the latches and obtained the initial clock period (column 8), which is larger than the clock period of the pulsed latch-based design (column 4), due to the larger sequencing overhead of the flip-flops. The optimum clock period is shown in column 9; this was obtained by clock skew scheduling [22] while assuming that an arbitrary amount of skew can be assigned. The clock period after clock skew scheduling, while allowing a skew of up to 10% of P_{opt}, which is the skew constraint used in our approach, is denoted by P_{css} and is shown in column 10; its FOM, η', which is defined similarly to (15) if we substitute P_{css} for P_{pwcs}, is shown in column 11. Comparing the optimum clock period of the two styles of circuit (columns 5 and 9) demonstrates the benefit of pulsed latch-based circuits that results from a reduced sequencing overhead (see Table I). Comparing P_{pwcs} and P_{css}, together with their corresponding FOMs, shows the advantage of pulsed latch-based circuits designed by combining clock skew scheduling and time borrowing by exploiting multiple pulse widths, against flip-flop-based circuits designed by clock skew scheduling alone.

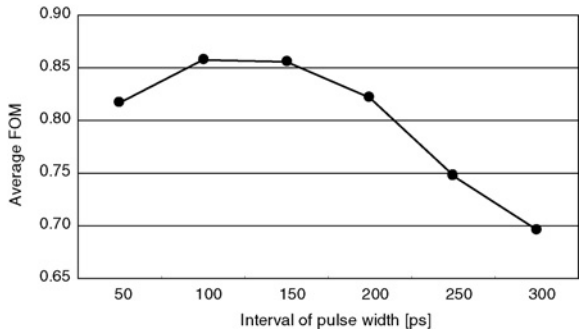


Fig. 10. Average FOM against pulse width interval.

B. Design Considerations of Pulse Generators

The list of available pulse widths $\mathcal{W}$ is important for *PWCS_Optimize* in its capability to reduce the clock period. We ran *PWCS_Optimize* for the benchmark circuits of Table III, while varying the pulse width interval between consecutive pulse generators (see Table II) from 50 ps to 300 ps in increments of 50 ps, with PG1 fixed to 156 ps, and obtained the average FOM as shown in Fig. 10. The total number of pulse generators available in a library will be limited in practice, and we have assumed this limit to be 5. As a result, too small an interval (e.g., 50 ps) restricts the maximum pulse width which in turn limits the maximum amount of time borrowing, and thus yields a lower average FOM. Conversely, too large an interval (e.g., 300 ps) forces an unnecessarily large pulse width to be selected (see L11 of Fig. 4), which is likely to limit the time borrowing of some other combinational blocks, reducing the average FOM. A pulse width interval of 100 ps turned out to yield the best average FOM, as shown in Fig. 10, and therefore this was used as the basis for designing the pulse generators shown in Table II.

Another approach is to use a different pulse width interval for each circuit. For example, s1423 has rather a large clock period, and so we may use 300 ps (about 10% of P_{ini}) to increase the amount of time borrowing. But this approach requires a large number of pulse generators to be present in the library; and pulse generators with wider pulses occupy more area and are more susceptible to process variations.

C. Physical Design

Fig. 11 compares the area of the two styles of circuits: the left-hand bars correspond to flip-flop-based circuits and show the proportions of the total area taken up by clock buffers, flip-flops, and combinational gates; the right-hand bars correspond to the pulsed latch-based circuits produced by following the design flow shown in Fig. 3 and show the proportions of extra buffers required to fix hold time violations, pulse generators, clock buffers, latches, and combinational gates, where numbers are normalized to the total area of the flip-flop-based circuits. Even though pulsed latch-based circuits involve extra buffers and pulse generators, the overall area occupied by the elements other than combinational gates is reduced because the area of a latch is smaller than that of a flip-flop (see Table I). The total area of pulsed latch-based circuits is reduced by 11%

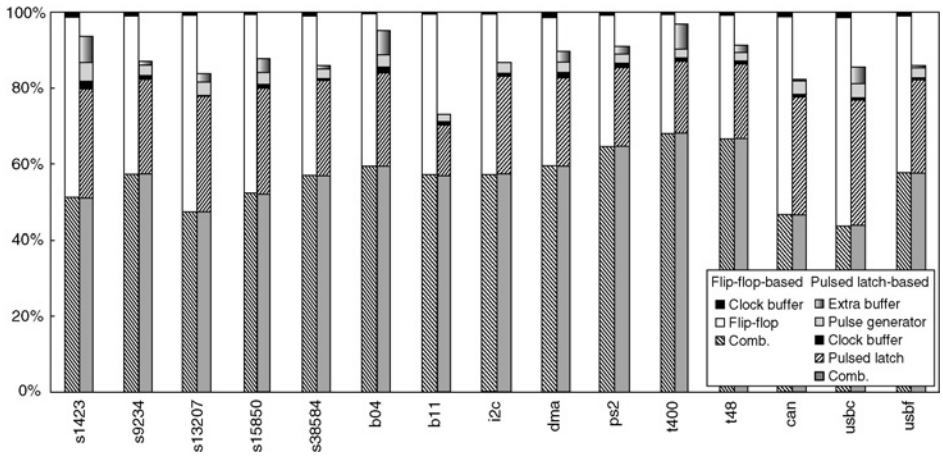


Fig. 11. Comparison of area between flip-flop-based circuits (left bars) and pulsed latch-based circuits (right bars), optimized with *PWCS_Optimize*. These results are normalized to the total area of the flip-flop-based circuits.

TABLE V

COMPARISON OF THE LENGTH OF SIGNAL WIRES, CLOCK WIRES, AND ALL WIRES, TOGETHER WITH AVERAGE CONGESTION, WHEN WE DO NOT ASSIGN A HIGHER NET WEIGHT TO THE NETS CONNECTING PULSE GENERATORS AND LATCHES ($W_n = 1$), AND WHEN WE DO FORCE A HIGHER NET WEIGHT ($W_n = 2$)

	Signal Wires (mm)			Clock Wires (mm)			Total Wires (mm)			Average Congestion (%)		
Name	$W_n = 1$	$W_n = 2$	Inc. (%)	$W_n = 1$	$W_n = 2$	Inc. (%)	$W_n = 1$	$W_n = 2$	Inc. (%)	$W_n = 1$	$W_n = 2$	Inc.
s1423	4.1	4.4	5.7	0.6	0.6	−3.1	4.7	4.9	4.6	11.4	12.1	0.7
s9234	11.3	12.7	12.4	1.9	1.7	−11.2	13.2	14.4	9.0	17.1	18.4	1.3
s13207	33.6	38.0	13.1	10.7	8.5	−21.0	44.3	46.5	4.9	21.4	22.1	0.7
s15850	41.3	43.9	6.2	10.3	7.8	−24.6	51.6	51.7	0.1	21.4	21.1	−0.2
s38584	168.2	183.4	9.0	60.1	49.7	−17.3	228.3	233.1	2.1	31.6	32.2	0.5
b04	4.9	5.2	7.3	0.5	0.4	−11.6	5.3	5.6	5.7	12.1	12.5	0.4
b11	4.3	4.4	1.5	0.3	0.2	−18.8	4.6	4.6	0.2	15.1	15.2	0.0
i2c	10.5	10.6	0.8	1.3	1.1	−17.3	11.8	11.7	−1.2	16.2	15.8	−0.4
dma	39.1	40.2	2.8	3.0	1.8	−41.5	42.1	41.9	−0.4	31.8	31.6	−0.2
ps2	20.6	21.7	5.4	2.2	1.5	−33.1	22.9	23.2	1.7	18.4	18.6	0.2
t400	28.9	30.1	4.1	1.7	1.4	−17.0	30.6	31.6	3.0	22.5	22.9	0.5
t48	38.7	42.1	8.8	3.6	2.8	−23.0	42.3	44.8	6.1	26.7	28.4	1.7
can	75.7	82.9	9.4	6.6	4.2	−36.1	82.3	87.1	5.8	24.9	26.0	1.1
usbc	27.3	28.1	3.1	3.2	2.0	−38.6	30.5	30.1	−1.3	19.1	18.7	−0.3
usbf	239.1	260.0	8.7	20.5	14.0	−31.8	259.7	274.0	5.5	29.8	31.3	1.5
Average			6.6			−23.1			3.0			0.5

on average, and the amount of reduction is determined by the proportion of sequencing elements: thus, the reduction is largest in b11 and smallest in t400.

As we discussed in Section III-E, a higher net weight is assigned to the nets between pulse generators and latches, so that they are located physically close after automatic placement. This, however, conflicts with the usual goal of placement to minimize the total wirelength (or average congestion), and thus the modified net weight may have a negative impact on the length of the signal wires. To assess the extent of this problem, we forced about 70% of the placement region to be occupied by cells in each case, which is a tight placement. We allowed metal layers up to M3 for routing. The placement region was divided into a grid of 1.6 μm × 1.6 μm squares to compute congestion.

In columns 2–4 of Table V, we compare the wirelength of signal wires when we do not assign a particular net weight, which is equivalent to assigning a net weight of 1, and when we force a higher net weight, which is 2 in the experiments with a router [27], on the nets between pulse generators and latches. The length of signal wires increases by 6.6% on average. However, the length of clock wires, which include all the wires between a clock source and latches, actually decreases, as reported in columns 5–7. This is understandable, because any particular pulse generator and the latches driven by it are forced into closer proximity, which reduces the length of the wires between them. The wirelength of total wires including both signal and clock wires is shown in columns 8–10; it increases 3.0% on average, but there are examples in which the wirelength decreases rather than increases. Average congestion is compared in the last three columns, which shows that the higher net weight has a negligible impact on congestion.

In Section III, we assumed that pulse width allocation and clock skew scheduling are performed before physical design; these process, however, can also be performed after

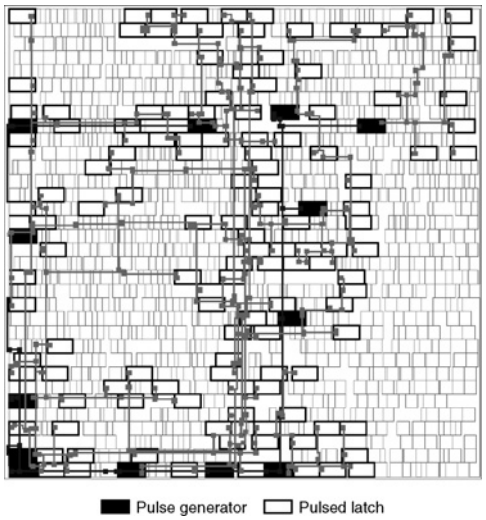


Fig. 12. Layout of `i2c` after allocating pulse generators to latches, placement and routing, and synthesis of clock trees.

placement. This has the advantage of allowing accurate wire loads and wire delays to be used, which in turn allows accurate maximum and minimum combinational delays, D_{ij} and d_{ij}, be used in (11) and (12), respectively. The drawback is that latch clustering has to be performed when latches have already been placed, which may require additional iterations of *PWCS_Optimize* and placement; this is left for further investigation. We instead checked five circuits (`s1423`, `s9234`, `s38584`, `dma`, and `usbc`) to see how the increase in the length of signal wires due to a higher net weight, as shown in Table V, affect the FOM. We extracted wire parasitics from the layout of each circuit for both $W_n = 1$ and $W_n = 2$. We annotated them back to netlist, and computed D_{ij} again. We then obtained new values of P_{ini}, P_{opt}, and P_{pwcs}; when $W_n = 2$, P_{ini} increased by 0.9% on average compared with P_{ini} for $W_n = 1$, P_{opt} decreased by 1.3%, and P_{pwcs} increased by 1.0%. This leads us to believe that the increased length of signal wires that results from the use of a higher net weight to cluster the pulse generator and latches has marginal effect.

Fig. 12 shows the final layout of an example circuit `i2c`, which was obtained by using the design flow in Fig. 3. It can be seen that the pulse generators are physically close to the latches driven by them, so that the delivery of pulses can be made local.

D. Power Analysis

We assessed the power consumption of pulsed latch-based circuits and compared it to that of flip-flop-based circuits using five examples (`s1423`, `s9234`, `s38584`, `dma`, and `usbc`). Fig. 13 illustrates the power consumption due to the combinational logic, the sequential elements, and the clock network. Random vectors were applied to the primary inputs every ten clock cycles, and thus the switching activity of combinational logic was kept below 10%.

The power consumption of a clock network increases mainly due to the pulse generators. This is especially true in smaller circuits such as `s1423` which only have a small number of clock buffers, so that the pulse generators in the clock network

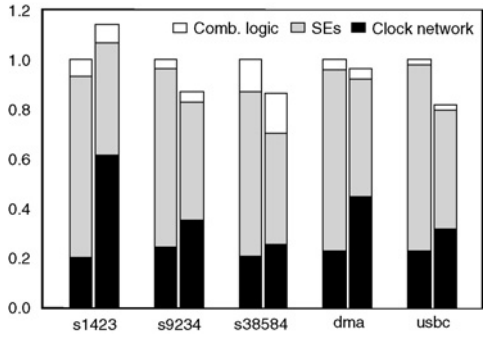


Fig. 13. Comparing power consumption of flip-flop-based circuits in the left bars and pulsed latch-based circuits in the right bars.

dominate its power consumption. The power consumption of sequential elements decreases due to less internal capacitance of latches. On average, the overall power consumption of pulsed latch-based circuits is 6.7% less than the flip-flop-based equivalents.

V. Conclusion

We have presented a pulsed latch-based design of sequential circuits, focusing on the primary problem of minimizing the clock period through pulse width allocation and clock skew scheduling. By combining a small number of different pulse widths with clock skews of up to 10% of the clock period, we showed through experiments that a minimum clock period can be achieved for many benchmark circuits, and a figure of merit of 0.86 on average. The algorithm for finding a minimum clock period *PWCS_Optimize* has been presented. The design flow, which consists of allocating pulsed latches to particular pulse generators, placement and routing, and synthesis of local and global clock trees, has also been presented and demonstrated. In spite of extra pulse generators, pulsed latch-based circuits have been shown to occupy 11% less area on average than their flip-flop-based counterparts.

There are a number of topics that are worth of further investigation. *PWCS_Optimize* could be performed after placement, which can reflect accurate wire loads and wire delays. But latches are then dispersed over the placement region, and thus may need more pulse generators. In-place optimization of pulse generators and latches would alleviate the problem. We currently rely on a higher net weight to force pulse generators and latches close during placement. Placement with the distance constraint would yield a better solution. We treat the latch timing parameters, namely T_{su}, T_{hd}, and T_{dq}, as precharacterized constants, which is a usual practice. Since they are dependent in theory [30], [31], considering them as variables allows room for further optimization, yet the problem becomes more complicated.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions.

REFERENCES

[1] D. Chinnery and K. Keutzer, "Introduction and overview of the book," in *Closing the Gap Between ASIC & Custom*. Norwell, MA: Kluwer, 2002, pp. 4–28.

[2] H. Partovi, R. Burd, U. Salim, F. Weber, L. DiGregorio, and D. Draper, "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, M. Daito, Y. Sugiyama, H. Suzuki, H. Morita, M. Nomura, S. I. K. Nadehara, M. Tokuda, Y. Inoue, T. Nakayama, H. Harigai, and Y. Yano, "A 100 MHz 0.4 W 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, M. Golden, N. Juffa, S. Meler, S. Oberman, H. Partovi, and F. Weber, "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. T. Clark, E. J. Hoffman, J. Miller, M. Biyani, L. Liao, S. Strazdus, M. Morrow, K. E. Velarde, and M. A. Yarch, "An embedded 32-b microprocessor core for low-power and high-performance applications," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1599–1608, Nov. 2001.

[6] N. A. Kurd, J. S. Barkarullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland, "A multigigahertz clocking scheme for the Pentium 4 microprocessor," *IEEE J. Solid-State Circuits*, vol. 36, no. 11, pp. 1647–1653, Nov. 2001.

[7] S. D. Naffziger, G. Colon-Bonet, T. Fischer, R. Riedlinger, T. J. Sullivan, and T. Grutkowski, "The implementation of the Itanium 2 microprocessor," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1448–1460, Nov. 2002.

[8] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita, and H. Sugiyama, "A 1.3 GHz fifth-generation SPARC64 microprocessor," *IEEE J. Solid-State Circuits*, vol. 38, no. 11, pp. 1896–1905, Nov. 2003.

[9] S. Shibatani and A. Li. (2006, Jul.). Pulse-latch approach reduces dynamic power, *EE Times* [Online]. Available: http://www.eetimes.com

[10] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in *Proc. CalTech Conf. Very-Large-Scale Integrat.*, Mar. 1983, pp. 23–36.

[11] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 945–951, Jul. 1990.

[12] G. Even, I. Y. Spillinger, and L. Stok, "Retiming revisited and reversed," *IEEE Trans. Comput.-Aided Design*, vol. 15, no. 3, pp. 348–357, Mar. 1996.

[13] K. Ravindran, A. Kuehlmann, and E. Sentovich, "Multi-domain clock skew scheduling," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2003, pp. 801–808.

[14] S. Sapatnekar, "Timing analysis for sequential circuits" in *Timing*. Norwell, MA: Kluwer, 2004, pp. 137–144.

[15] S. Sapatnekar, "Clocking and clock skew optimization," in *Timing*. Norwell, MA: Kluwer, 2004, pp. 190–196.

[16] C. Chiang and J. Kawa, "Design for yield," in *Design for Manufacturability and Yield for Nano-Scale CMOS*. Berlin, Germany: Springer, 2007, p. 173.

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

[18] S. Held, B. Korte, J. Maßberg, M. Ringe, and J. Vygen, "Clock scheduling and clocktree construction for high-performance ASICs," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2003, pp. 232–239.

[19] P. Restle, T. McNamara, D. Webber, P. Camporese, K. Eng, K. Jenkins, D. Allen, M. Rohn, M. Quaranta, D. Boerstler, C. Alpert, C. Carter, R. Bailey, J. Petrovick, B. Krauter, and B. McCredie, "A clock distribution network for microprocessors," *IEEE J. Solid-State Circuits*, vol. 36, no. 5, pp. 792–799, May 2001.

[20] R. Kumar, K. Bollapalli, R. Garg, T. Soni, and S. Khatri, "A robust pulsed flip-flop and its use in enhanced scan design," in *Proc. Int. Conf. Comput. Design*, Oct. 2009, pp. 97–102.

[21] S. Unger and C. Tan, "Clocking schemes for high-speed digital systems," *IEEE Trans. Comput.*, vol. 35, no. 10, pp. 880–895, Oct. 1986.

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

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

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

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

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

[27] *Astro User Guide*, Synopsys, Inc., Mountain View, CA, Jun. 2006.

[28] *Opencores* [Online]. Available: http://www.opencores.org/

[29] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, Tech. Rep. UCB/ERL M92/41, May 1992.

[30] E. Salman, A. Dasdan, F. Taraporevala, K. Kucukcakar, and E. G. Friedman, "Exploiting setup/hold-time interdependence in static timing analysis," *IEEE Trans. Comput.-Aided Design*, vol. 26, no. 6, pp. 1114–1125, Jun. 2007.

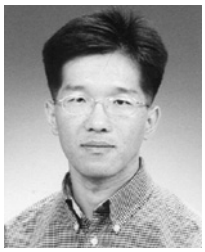
[31] S. Srivastava and J. Roychowdhury, "Independent and interdependent latch setup/hold time characterization via Newton–Raphson solution and Euler curve tracking of state-transition equations," *IEEE Trans. Comput.-Aided Design*, vol. 27, no. 5, pp. 817–830, May 2008.

Hyein Lee received the B.S. degree in electronic engineering from Yonsei University, Seoul, Korea, in 2007, and the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2009.

She is currently with Design Technology Team, Samsung Electronics, Yongin, Korea. Her research interests include VLSI design methodology and computer-aided design for high-performance integrated circuits.

Seungwhun Paik (S'07) received the B.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2006. He is currently working toward the Ph.D. degree at the Department of Electrical Engineering, KAIST.

His research interests are in the areas of computer-aided design for high-performance designs, low power designs, high-level synthesis, and structured ASIC.



Youngsoo Shin (M'00–SM'05) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, Korea. He received the Ph.D. degree in electronics engineering from Seoul National University in 2000.

From 2000 to 2001, he was with the University of Tokyo, Japan, as a Research Associate, and from 2001 to 2004, he was with the IBM T.J. Watson Research Center, Yorktown Heights, NY, as a Research Staff Member. He has been with the Department of Electrical Engineering, KAIST, Daejeon, Korea, since 2004, where he is currently an Associate Professor. His research interests include the areas of computer-aided design with emphasis on low-power design and design tools, high-level synthesis, sequential synthesis, and structured application-specific integrated circuits.

Dr. Shin received the Best Paper Award at the 2005 International Symposium on Quality Electronic Design, and was nominated for the Best Paper Award at the same conference in 2007. He has been a member of the technical program committee and organizing committee of several technical conferences, including the Design Automation Conference, the International Conference on Computer Aided Design, the International Symposium on Low Power Electronics and Design, the Asia and South Pacific Design Automation Conference, the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, the IEEE Computer Society Annual Symposium on Very Large Scale Integration, and the International Symposium on Circuits and Systems.