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 VDD. 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 clock period, and the average reduction in area by 11%. The average figure of merit of 0.86, where 1.0 indicates a minimum clock period for many benchmark circuits. The results have an 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

F LIP-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. 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.

1Ideally, 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.
TABLE I

<table>
<thead>
<tr>
<th></th>
<th>F/F</th>
<th>Latch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup time (ps)</td>
<td>72</td>
<td>47</td>
</tr>
<tr>
<td>Clock-to-Q delay (ps)</td>
<td>238</td>
<td>190</td>
</tr>
<tr>
<td>Clock load (fF)</td>
<td>3.4</td>
<td>2.2</td>
</tr>
<tr>
<td>Area (µm²)</td>
<td>8.0</td>
<td>4.8</td>
</tr>
</tbody>
</table>

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 skew [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 skew 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.
II. TIMING CONSTRAINTS OF SEQUENTIAL CIRCUITS

In this section, we review the timing constraints of sequential circuits, setup and hold 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 $T_{thd}$, 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 $T_{su}$ and $T_{thd}$ 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

$$T_{su} + D_{ij} \leq P - T_{su}. \quad (1)$$

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

$$T_{thd} + d_{ij} \geq T_{thd}. \quad (2)$$

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]

$$A_i = \max \left( T_{cq}, A_{i-1} + T_{dQ} \right) \quad (3)$$

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

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

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

$$a_i = \min \left( T_{cq}, a_{i-1} + T_{dQ} \right) + d_{ij} \quad (5)$$

then the hold time constraint can be described by

$$\max \left( T_{cq}, a_i + T_{dQ} \right) + d_{ij} \geq W + T_{thd}. \quad (6)$$
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 \( T_{\text{su}} \) 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 \( W - T_{\text{su}} \), which is the latest time at which data can depart from \( i \), and \( P + W - T_{\text{su}} \), which is the latest time at which data can arrive to \( j \) [see Fig. 2(c)]. This yields the setup time constraint

\[
(W - T_{\text{su}}) + T_{\text{su}} + D_j \leq P + (W - T_{\text{su}}). \tag{7}
\]

Note that this constraint is similar to (1) if we remove \( W - T_{\text{su}} \) 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

\[
T_{\text{su}} + d_j \geq W + T_{\text{su}}. \tag{8}
\]

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.

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

\[
S_i + T_{\text{su}} + D_j \leq P + S_j, \tag{9}
\]

\[
S_i + T_{\text{su}} + d_j \geq S_j + W + T_{\text{su}}. \tag{10}
\]

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 \( S_i \). The minimum \( P \), under the assumption that \( S_i \) takes an arbitrary value, can be derived by verifying (9) and (10) log \( N \) times [22], where \( N \) is the number of potential clock periods that can be tried. If the values of \( S_i \) 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 \( W_i \in W \), where \( W \) is a list of

2This 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

$$S_i + W_i + T_{dH} + D_{ij} \leq P + S_j + W_j$$  \hspace{0.5cm} (11)

$$S_i + T_{dH} + D_{ij} \geq S_j + W_j + T_{dL}$$  \hspace{0.5cm} (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 $W_i$ and a maximum allowable clock skew $\Delta$, the PWCS optimization problem is to allocate a pulse width $W_i \in W_i$ 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 $\log_2 (P_{\text{max}} - P_{\text{min}})/\epsilon$ tries, where $P_{\text{max}}$ and $P_{\text{min}}$, respectively, are the upper and lower bounds of the clock period that can be tried and $\epsilon$ is an increment of the clock period, e.g., 1 ps. The upper bound $P_{\text{max}}$ can be derived from (11) by setting all pulse widths equal and all skews to 0 [22]

$$P_{\text{max}} = \max\{T_{dH} + D_{ij}\}.$$  \hspace{0.5cm} (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 $P_{\text{min}}$

$$P_{\text{min}} = \max \{\min\limits_{i \neq j} \{T_{dL} + D_{ij}\}, \max\limits_{i \neq j} \{T_{dH} + D_{ij}\}\}.$$  \hspace{0.5cm} (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 $W_i$ and a maximum allowable clock skew $\Delta$, the PWCS problem is to find $W_i \in W_i$ and $S_i \leq \Delta$ such that a specified clock period $P$ can be satisfied under (11) and (12).

D. Algorithm

The algorithm PWCS Optimizer, which we use to solve Problem 1, is shown in Fig. 4. It iteratively checks the median of the clock period (L3) between the current maximum $P_c$ 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 Optimizer ($P_{\text{max}}, P_{\text{min}}, \epsilon$)**

begin
1. $P_c := P_{\text{max}}$, $P_l := P_{\text{min}}$
2. while $(P_c - P_l) > \epsilon$ do
3. $P = (P_c + P_l)/2$
4. if PWCS($P$) = success then $P_c := P$ else $P_l := P$
end if
end do
end

**Function PWCS($P$)**

begin
6. Set a list of pulse widths $W_i$ in order of increasing width
7. $W_i := \min_{i \in C}(W_i)$, $S_i := 0$, $\forall i$
8. $C(i, j) = W_i + S_i + T_{dH} + D_{ij} - P$ \hspace{0.5cm} (11)
9. while $C(i, j)$ is not satisfied do
10. $r_{ij} := W_i + S_i + T_{dH} + D_{ij} - P$
11. Increment $W_i$ until $W_i \geq r_{ij}$
12. if such $W_i$ in $W_i$ does not exit then return fail
end if
13. $S_i := \max\{0, r_{ij} - W_i\}$
end do
end return success
end

![Fig. 4. Pseudocld of the PWCS optimization algorithm.](image)

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 ($W_i$) 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 $r_{ij}$, is considered to be fixed (L10), and we regard the parameters $W_i$ 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 Optimizer calls PWCS $\log_2 (P_{\text{max}} - P_{\text{min}})/\epsilon$ times, thus the bound on
PWCS_optimize is $O(n \log_2(P_{max} - P_{min})$/\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 $O(n)$. We now need to determine values of $W_j$ and $S_j$ with a sum that is not smaller than $r_j$ (L11). We want to make this sum as small as possible so that it is larger than or equal to $r_j$, so that $W_j$ and $S_j$ become less restrictive when $j$ is located on the right-hand side ($r_j$) in later iterations. Since $W_j$ takes one of the discrete number of values in $\V_e$, while $S_j$ takes any value between 0 and $\Delta$, 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 ($\Delta$) 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 $T_{max}$ is 1 time unit, that the maximum allowable skew ($\Delta$) is 3 time units, and that the pulse widths can be selected from $\V_e = \{5, 10\}$. 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 $a \sim b$ and $b \sim c$ violate the setup time constraint (11). Suppose that we select $b \sim c$ for the first iteration (L9). As shown in Fig. 5(b), $r_3$ is 6 time units ($D_{ij}$), which yields $W_3 = 5$ (L11): note that $W_f + \Delta \geq 6$ returns $W_f = 5$ for $\Delta = 3$ and $S_i = 1$ (L13). Further suppose that we select $a \sim b$ for the second iteration, which yields $W_4 = 10$ and $S_3 = 1$; this causes $b \sim c$ 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.

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 $\rho_1$ (L2). We then subtract $\rho_1$ from the skews of all the latches in the group (L3); the value of $\rho_1$ 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. 5. (a) Example sequential circuit. (b) Iterations of PWCS.

Algorithm Clock_Tree_Synthesis ($\phi_1, \phi_2, \ldots, \phi_n$)
begin
L1 for $i = 1, 2, \ldots, n$ do
L2 $\rho_i := \min_{\phi_j \in \phi_i} S_j$
L3 $S_j := S_j - \rho_i \forall \phi_j$
L4 Local clock tree synthesis of $\phi_i$
end do
L5 Adjust_PG_Skew ()
L6 Global clock tree synthesis of $\phi_1, \phi_2, \ldots, \phi_n$
end

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

Fig. 6. Pseudocode of clock tree synthesis algorithm.
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.

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 \( \rho_i \) (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, \( \rho_2 = S_2 = 2 \) is subtracted from all skews in the second group. After synthesis of the two local clock trees, the delay between PG and latch 1 is the smallest in \( \phi_1 \), and \( \phi_2 \) has the smallest delay between PG 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 (\( \rho_2 \)) 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 (usb2), 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.

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...
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 skew 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 in Reducing the Clock Period

The results of PWCS 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 $P_{\text{ini}}$ and is shown in column 4. $P_{\text{ini}}$ 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, followed by a process of fixing hold time violations, is denoted by $P_{\text{pwcs}}$ and is shown in column 6. The maximum allowable skew ($\Delta t$) was assumed to be 10% of $P_{\text{ini}}$ [13], [17]. Out of five pulse generators in Table II, only those with pulse widths within 40% of $P_{\text{ini}}$ 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 $P_{\text{ini}}$. In order to assess the effectiveness of PWCS 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 PWCS.

$$\eta = \frac{P_{\text{ini}} - P_{\text{pwcs}}}{P_{\text{ini}}}$$

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

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 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 column 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 ($s1423$, $s13207$, $s15850$, s14010, s148, and can). Three circuits ($s13207$, ps2, and can) have rather a small $\eta$, below 0.8, even before the hold time violations are fixed. This is because the maximum skew value used in these designs to obtain $P_{\text{ini}}$ is considerably larger than the maximum clock skew and pulse width that PWCS can use, which is a skew up to 10% of $P_{\text{ini}}$ 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 $P_{\text{ini}}$, unlike the other circuits.

The distribution of pulse generators after running PWCS is shown in Fig. 9, which demonstrates the maldistribution of PG1, the pulse generator with the narrowest pulse width. If all the pulse generators were restricted to PG1 alone, the average $\eta$ would be 0.51; thus, even though PG1 dominates in numbers, exploiting the small number of pulse generators with wider pulse widths allows a significant reduction of the clock period, resulting in the average $\eta$ 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

<table>
<thead>
<tr>
<th>Name</th>
<th>FOM (m)</th>
<th># of Hold Time Constraints</th>
<th># of Hold Time Violations</th>
<th># of Extra Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1423</td>
<td>1.00</td>
<td>0.97</td>
<td>248956</td>
<td>230</td>
</tr>
<tr>
<td>s5234</td>
<td>1.00</td>
<td>1.00</td>
<td>206088</td>
<td>24</td>
</tr>
<tr>
<td>s132077</td>
<td>0.97</td>
<td>0.97</td>
<td>2039024</td>
<td>146</td>
</tr>
<tr>
<td>s15850</td>
<td>0.98</td>
<td>0.75</td>
<td>4532317</td>
<td>340</td>
</tr>
<tr>
<td>s183504</td>
<td>1.00</td>
<td>1.00</td>
<td>17357304</td>
<td>165</td>
</tr>
<tr>
<td>s101</td>
<td>0.90</td>
<td>0.90</td>
<td>69798</td>
<td>34</td>
</tr>
<tr>
<td>s21</td>
<td>1.00</td>
<td>1.00</td>
<td>14832</td>
<td>4</td>
</tr>
<tr>
<td>s15</td>
<td>1.00</td>
<td>1.00</td>
<td>14832</td>
<td>4</td>
</tr>
<tr>
<td>s1423</td>
<td>1.00</td>
<td>1.00</td>
<td>14832</td>
<td>4</td>
</tr>
<tr>
<td>s101</td>
<td>0.90</td>
<td>0.90</td>
<td>69798</td>
<td>34</td>
</tr>
<tr>
<td>s21</td>
<td>1.00</td>
<td>1.00</td>
<td>14832</td>
<td>4</td>
</tr>
<tr>
<td>s15</td>
<td>1.00</td>
<td>1.00</td>
<td>14832</td>
<td>4</td>
</tr>
<tr>
<td>s1423</td>
<td>1.00</td>
<td>0.76</td>
<td>148322</td>
<td>144</td>
</tr>
<tr>
<td>s1423</td>
<td>1.00</td>
<td>0.80</td>
<td>482182</td>
<td>70</td>
</tr>
<tr>
<td>s1423</td>
<td>0.80</td>
<td>0.58</td>
<td>1405900</td>
<td>41</td>
</tr>
<tr>
<td>s1423</td>
<td>0.80</td>
<td>0.80</td>
<td>1405900</td>
<td>41</td>
</tr>
<tr>
<td>Average</td>
<td>0.92</td>
<td>0.86</td>
<td>1451524</td>
<td>524</td>
</tr>
</tbody>
</table>

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\textsubscript{opt}, which is the skew constraint used in
our approach, is denoted by P\textsubscript{css} and is shown in column 10; its FOM, \( \eta \),
which is defined similarly to (15) if we substitute P\textsubscript{css} for P\textsubscript{PWS},
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\textsubscript{PWS} and P\textsubscript{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.

B. Design Considerations of Pulse Generators

The list of available pulse widths \( W \) is important for PWCS_optimize in its capability to reduce the clock pe-
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 \( P_0 \))
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%
on average, and the amount of reduction is determined by the proportion of sequencing elements; thus, the reduction is largest in b1.1 and smallest in t440.

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 congestion is compared in the last three columns, which shows that the higher net weight has a negligible impact on 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

![Table V](image)

**Table V**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal Wires (mm)</th>
<th>Clock Wires (mm)</th>
<th>Total Wires (mm)</th>
<th>Average Congestion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wn</td>
<td>Wn</td>
<td>Inc. (%)</td>
<td>Wn</td>
</tr>
<tr>
<td>t443</td>
<td>4.1</td>
<td>4.4</td>
<td>3.1</td>
<td>4.7</td>
</tr>
<tr>
<td>t2534</td>
<td>11.3</td>
<td>12.7</td>
<td>1.1</td>
<td>12.2</td>
</tr>
<tr>
<td>t2530</td>
<td>38.6</td>
<td>39.9</td>
<td>0.3</td>
<td>39.5</td>
</tr>
<tr>
<td>t35850</td>
<td>41.3</td>
<td>43.9</td>
<td>6.2</td>
<td>43.5</td>
</tr>
<tr>
<td>t35854</td>
<td>168.2</td>
<td>183.4</td>
<td>9.0</td>
<td>182.5</td>
</tr>
<tr>
<td>t51</td>
<td>3.3</td>
<td>4.4</td>
<td>3.5</td>
<td>4.6</td>
</tr>
<tr>
<td>t258</td>
<td>10.6</td>
<td>11.6</td>
<td>9.8</td>
<td>11.3</td>
</tr>
<tr>
<td>t358</td>
<td>56.7</td>
<td>66.5</td>
<td>1.4</td>
<td>66.4</td>
</tr>
<tr>
<td>t456</td>
<td>28.7</td>
<td>35.2</td>
<td>2.4</td>
<td>35.1</td>
</tr>
<tr>
<td>t440</td>
<td>38.7</td>
<td>42.1</td>
<td>8.8</td>
<td>42.0</td>
</tr>
<tr>
<td>t31</td>
<td>75.7</td>
<td>82.9</td>
<td>9.4</td>
<td>82.5</td>
</tr>
<tr>
<td>t233</td>
<td>20.7</td>
<td>26.4</td>
<td>2.7</td>
<td>26.6</td>
</tr>
<tr>
<td>t383</td>
<td>20.1</td>
<td>26.0</td>
<td>3.1</td>
<td>26.0</td>
</tr>
<tr>
<td>Average</td>
<td>6.6</td>
<td>−3.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

**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 (Wn = 1), and When We Do Force a Higher Net Weight (Wn = 2)**

Authorized licensed use limited to: Korea Advanced Institute of Science and Technology. Downloaded on February 23,2010 at 01:10:30 EST from IEEE Xplore. Restrictions apply.
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, $D_j$ and $d_j$, to 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 $\text{Optimize}$ and placement; this is left for further investigation. We instead checked five circuits ($s_{1423}$, $s_{9234}$, $s_{38584}$, dma, and $\text{usbc}$) 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 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 ($s_{1423}$, $s_{9234}$, $s_{38584}$, dma, and $\text{usbc}$). Fig. 13 illustrates the power consumption due to the combination 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 $s_{1423}$ which only have a small number of clock buffers, so that the pulse generators in the clock network 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 $\text{PWCS}_{\text{Optimize}}$ 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 $\text{Optimize}$ 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 $T_{\text{ss}}$, $T_{\text{s}}$, and $T_{\text{q}}$, 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.