Pulse Width Allocation with Clock Skew Scheduling for Optimizing Pulsed Latch-Based Sequential Circuits

Hyein Lee, Seungwhun Paik, and Youngsoo Shin
Department of Electrical Engineering, KAIST
Daejeon 305-701, Korea

Abstract— **Pulsed latches, latches driven by a brief clock pulse, offer the convenience of flip-flop-like timing verification and optimization, while retaining superior design parameters of latches over flip-flops. But, pulsed latch-based design using a single pulse width has a limitation in reducing clock period. The limitation still exists even if clock skew scheduling is employed, since the amount of skew that can be assigned is practically limited due to process variations. The problem of allocating pulse width (out of discrete number of predefined widths) and scheduling clock skew (within prescribed upper bound) is formulated, for the first time, for optimizing pulsed latch-based sequential circuits. An allocation algorithm called *PWCS_Optimize* is proposed to solve the problem. Experiments with 65-nm technology demonstrate that small number of variety of pulse widths (up to 5) combined with clock skews (up to 10% of clock period) yield minimum clock period for many benchmark circuits. The design flow including *PWCS_Optimize*, placement and routing, and synthesis of local and global clock trees is presented and assessed with example circuits.**

I. Introduction

Flip-flops are commonly used sequencing elements to design sequential circuits. In particular, edge-triggered sequential circuits, which consist of combinational blocks that lie between D flip-flops, are the most common form of sequential circuits in ASIC designs due to their convenience of timing verification. Flip-flops, however, impose significant overhead in terms of delay (setup time and clock-to-Q delay), clock load, and area than latches do. This is unavoidable since flip-flops are typically constructed by connecting two level-sensitive latches in a master-slave fashion.

Latches are therefore superior to flip-flops in terms of overhead of sequencing elements. Level-sensitive sequential circuits based on latches, nevertheless, make timing verification very difficult, since combinational blocks are not isolated each other due to transparent nature of latches. On the other hand, this transparency offers more flexibility in design, which is why they are widely used in high-performance microprocessors.

Pulsed latches are latches driven by a brief clock pulse. They retain the advantage of latches of superior design parameters, while offering flip-flop-like timing verification and optimization due to short period of transparency. Several types of pulsed latches have been proposed, mainly for high-performance microprocessors [1]–[4]; the application to ASICs have been reported [5] only recently. The clock pulse is internally generated by pulsed latch itself [1], or is generated by external pulse generators [2], [4], which receive a normal clock and generate a clock pulse that is then delivered to local pulsed latches.

Pulsed latch-based circuit using a single pulse width, which is a conventional model, virtually does not exploit time borrowing due to short period of transparency. It therefore has a limitation in reducing clock period for higher frequency or for lower V_{dd}. This can be alleviated by employing clock skew scheduling. However, conventional clock skew scheduling with arbitrary amount of skews has become impractical because within-die variation, which grows with technology scaling, affects extra buffers and wires (inserted for implementing large skews) in randomly different amount, thereby causing uncertainties in skews [6].

In this paper, we formulate the problem of allocating pulse widths (each width defined by a unique pulse generator) and scheduling clock skews (within upper bound, defined as a proportion of clock period) to minimize clock period of pulsed latch-based circuits. An algorithm called *PWCS_Optimize* is proposed to solve the problem. Experiments with 65-nm technology demonstrate that small number of various pulse widths (up to 5) combined with clock skews (up to 10% of clock period) yield figure of merit of 0.93 on average (1.0 indicates minimum clock period) for several benchmark circuits.

The pulsed latches with the same pulse width, determined by *PWCS_Optimize*, are grouped together so that they are driven by the same pulse generator, which is then followed by synthesis of local and global clock trees to realize the skews. The design flow down to the final circuit layout is presented and assessed, showing that area is reduced by 12% on average compared to flip-flop-based circuits, due to less area occupied by sequencing elements.

II. Timing Constraints of Sequential Circuits

In this section, we review the timing constraints (setup and hold time) of sequential circuits based on flip-flops, latches, and pulsed latches, with the help of Fig. 1. The setup time is denoted by T_{su}, hold time by T_{hd}, clock-to-Q delay by T_{cq}, data-to-Q delay by T_{dq}, and cycle time by P. We do not distinguish between propagation and contamination delay (i.e. maximum and minimum delay) of T_{cq} and T_{dq}, respectively, for simplicity of presentation. The timing parameters of sequencing elements in each type of circuit are assumed to be equal. The maximum delay of combinational block between sequencing elements i and j is denoted by D_{ij}, and the minimum delay by d_{ij}.

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

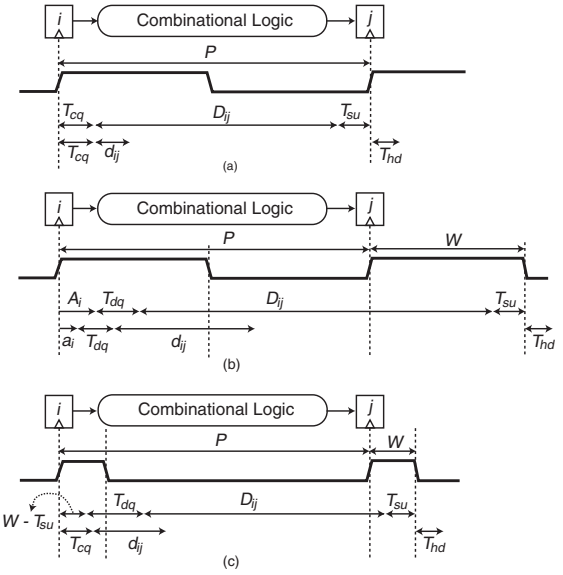

Fig. 1. Setup and hold time constraints: (a) flip-flop-based circuits, (b) latch-based circuits, and (c) pulsed latch-based circuits.

A. Flip-Flop-Based Circuits

In edge-triggered sequential circuits, data is launched from flip-flop i at the rising edge of clock (for positive edge-triggered) and the latest result of computation from combinational block has to arrive at flip-flop j earlier than setup time before the next rising edge of clock (see Fig. 1(a)), which constitutes the setup time constraint:

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

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

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

B. Latch-Based Circuits

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

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

where T_{cq} corresponds to data arriving at k before rising edge and $A_k + T_{dq}$ after rising edge. The setup time constraint between latches i and j can then be described [7] 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 clock being high (see Fig. 1(b)). Note that 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

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

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)) get 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 that use less than a clock period. The timing constraints will become similar to those of flip-flops ((1) and (2)) if we do not allow time borrowing, i.e. if we enforce all the combinational block to use no more than a clock period, but this will get rid of the benefit of latches on high-performance designs.

Since the amount of time borrowing is determined by W and pulsed latches are latches driven by a clock of very short W, we can safely choose to use flip-flop-like timing constraints (by preventing time borrowing) for the sake of convenience of setting up timing constraints. Let us assume that T_{su} is smaller than pulse width W, as shown in Fig. 1(c), which is usually true as we will see in Sec. IV. Each combinational block is allocated a clock period between $W - T_{su}$, the latest time data can depart at from i, and $P + W - T_{su}$, the latest time data can arrive at to j (see Fig. 1(c)), which yields, as the setup time constraint,

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

Note that (7) is similar to (1) after removing $W - T_{su}$ from both sides of inequality. The earliest time data can depart at is rising edge of clock, which leads to the hold time constraint:

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

III. Pulse Width Allocation with Clock Skew Scheduling

A. Overview

Pulsed latch-based circuits based on timing constraints (7) and (8), where we assumed the same pulse width W for all latches, do not have advantages over flip-flop-based circuits, except that we use sequencing elements of superior design parameters. However, if we allow a variety of different pulse widths to choose from and an intentional clock skew to assign for each pulsed latch, which is our model of pulsed latch-based circuits, there is a potential to reduce clock period.

Our approach to optimizing pulsed latch-based sequential circuits is illustrated in Fig. 2. We receive a gate-level netlist of a circuit synthesized with initial timing constraints including clock period, as an input to our pulse width allocation and clock skew scheduling (PWCS) optimization problem. A list of available pulse widths is defined by a library of pulse generators; clock skews are restricted to within a specified upper bound.

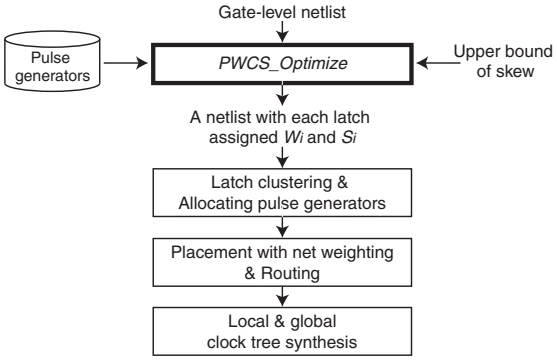

Fig. 2. Overall flow of pulse width allocation with clock skew scheduling.

Once pulsed latches are assigned pulse width and clock skew, the latches with the same pulse width are grouped (to an extent of the maximum number of latches that can be driven by a single pulse generator) and assigned to a specific pulse generator. The nets between each group of latches and pulse generator are assigned a higher net weight, so that they are placed close during automatic placement that follows. After automatic routing is performed, the skew assigned to each latch is realized through a synthesis of local clock tree [8] between pulse generator and a latch. The global clock tree between pulse generators and a clock source is then synthesized.

B. Timing Constraints for Multiple Pulse Widths and Clock Skew

If we assume that we can assign an intentional clock skew, denoted by S_i, for each latch i, the original setup time and hold time constraints ((7) and (8)) are modified to

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

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

For a specified P, (9) and (10) can be verified in polynomial time [9] to check whether there is a set of feasible S_is. The optimum P can be derived by verifying (9) and (10) $log_2 N$ times [9], where N is the number of potential clock periods, but only if arbitrary amount of skews can be assigned. If S_is are restricted (e.g. to 10% of clock period [6]), minimum P that can be achieved may be far from optimum.

To overcome this limitation, we further assume that we can allocate a different pulse width W_i to each latch i, which yields 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 introducing W_i and W_j in (11) has an effect of reinforcing time borrowing, since $W_i < W_j$ implies that combinational block between i and j can use more than a clock period.

C. Problem Formulation

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

Problem 1 Given a netlist of sequential circuit with timing constraints (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 pulse width $W_i \in \mathcal{W}$ and assign clock skew $S_i \leq \Delta$ to each pulsed latch i, with the objective of minimizing clock period while satisfying the setup time and hold time constraints as described by (11) and (12).

Note that Problem 1 can be solved by verifying iteratively (through binary search) whether we can find a feasible set of (W_i, S_i) for a specified clock period by $log_2 \left(P_{max} - P_{min}\right) / \epsilon$ times, where P_{max} and P_{min} are upper and lower bounds of minimum clock period, respectively, and ϵ is increment of clock period [9]. The upper bound P_{max} can be derived from (11) by setting all pulse widths equal and all skews 0 [9]:

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

The longest path from any latch to itself can serve as lower bound; the shortest path from any latch to some other latch also serves as lower bound [9]; maximum of these two is equal to 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 binary search is defined as PWCS problem that verifies whether feasible set of (W_i, S_i) exists for a specified 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 corresponds to solving Problem 1, is presented in Fig. 3. It iteratively checks a median period (L3) between current maximum P_u and minimum P_l through calling 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).

The function *PWCS*, which corresponds to solving Problem 2, is also presented in Fig. 3. In the implementation of this function, we consider the setup time constraint (11) alone, and ignore the hold time constraint (12). Once clock period is determined, any logic path that violates hold time constraint can be fixed by using an algorithm such as minimum padding [10] without affecting the determined clock period.

The function *PWCS* works as follows. A list of available pulse widths ($\mathcal{W}$) is arranged (L6) in order of increasing width. All latches are initialized to have minimum pulse width and 0 clock skew (L7). For each pair of latches i and j, setup time constraint (11) is constructed (L8), which is denoted by $C(i, j)$ in the algorithm. We then iterate a loop for each

Algorithm *PWCS_Optimize*$(P_{max}, P_{min}, \epsilon)$
begin
L1 $P_u := P_{max}, \quad P_l := P_{min}$
L2 **while** $(P_u - P_l) > \epsilon$ **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 $\mathcal{W}$ in order of increasing width
L7 $W_i := \min_{x \in \mathcal{W}} x, \; S_i := 0, \quad \forall_i$
L8 $C(i,j)$: $W_j + S_j \geq W_i + S_i + (T_{dq} + D_{ij} - P), \forall_{i \rightsquigarrow j}$
L9 **while** $\exists_{i,j}, C(i,j)$ is not satisfied **do**
L10 $r_h := W_i + S_i + (T_{dq} + D_{ij} - P)$
L11 Increment W_j until $W_j + \Delta \geq r_h$
L12 **if** such $W_j \in \mathcal{W}$ does not exist **then** **return** fail
end if
L13 $S_j := \max(0, r_h - W_j)$
end do
L14 **return** success
end

Fig. 3. Pseudo-code of PWCS optimization algorithm.

unsatisfied constraint $C(i,j)$ (L9). The right-hand side of inequality $C(i,j)$, denoted by r_h, is considered to be fixed (L10); we regard W_j and S_j, parameters of data-capturing latch, as variables. This is essentially iterative relaxation based approach for Bellman-Ford algorithm, which is proved to be optimal [11] in a sense that P is feasible if and only if *PWCS* returns in success.

We now want to determine W_j and S_j, such that their sum is not less than r_h (L11). Note that we want to make their 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 at right-hand side r_h in later iterations. Since W_j takes a discrete number of values in $\mathcal{W}$, while S_j takes any value between 0 and Δ, there can be discrete number of combinations of (W_j, S_j) that satisfy $C(i,j)$. We prefer the combination with smaller W_j, since the overhead of pulse generator increases with increasing pulse width it generates, which is made clear in Section IV. Therefore, we first assume maximum skew (Δ) and find the smallest possible pulse width that satisfies the constraint (L11). If such pulse width does not exist, $C(i,j)$ is not feasible, and thus we return in failure (L12). The skew S_j is then snapped to its smallest value (L13). Observe that some $C(i,j)$s that are made satisfied at the end of iterations can get violated again in later iterations, which then need to be processed again.

E. Latch Clustering and Physical Design

Once pulse width and skew are determined for each pulsed latch via algorithm *PWCS_Optimize*, the latches with the same pulse width have to be grouped, so that they can be driven by the same pulse generator (see the second step of Fig. 2). Since there is an upper bound on the number of latches that can be driven by a single pulse generator (10 in our experiments in Section IV), the latches with the same pulse width, if their numbers exceed this bound, are equally distributed to several pulse generators of equal pulse width.

While we group latches, we take their skews into account. All the latches with the same pulse width are initially arranged in order of increasing skew. When we distribute them to several pulse generators, we keep this order, so that latches of similar skews can be driven by the same pulse generator. In each group, the same amount of skew is subtracted from each latch, which is then realized as skew of pulse generator during global clock tree synthesis (the last step of Fig. 2). This is because, to realize excessive skew in local clock network (between pulse generator and latches), we may need long wires or many buffers, which can cause unreliable delivery of pulses.

Once we group latches, we assign higher net weight (3 in our experiments) on the nets between latches and pulse generator in the same group, so that they can be physically close during automatic placement (the third step of Fig. 2). This is followed by synthesis of local clock tree to realize skews assigned to latches, and by synthesis of global clock tree to realize skews assigned to pulse generators.

TABLE II
PULSE GENERATORS USED IN THE EXPERIMENT

Name	Pulse width (ps)	Area (μm^2)
PG1	131	5.12
PG2	228	5.44
PG3	334	6.08
PG4	432	6.40
PG5	532	7.04

IV. EXPERIMENTAL RESULTS

We carried out experiments on a set of sequential circuits taken from the ISCAS and ITC benchmarks. We also included circuits extracted from several open cores [12] including communication controller (`i2c`), keyboard interface (`ps2`), ECC core (`ecc`), and USB core (`usbf`). The first three columns of Table I give the name, the number of combinational gates, and the number of sequencing elements of each circuit.

Each circuit was synthesized with SIS [13]; a gate library used for technology mapping during the synthesis was constructed for 114 gates, which we based on 65-nm commercial technology. The netlist thus obtained is then submitted to algorithm *PWCS_Optimize*, which, together with function *PWCS*, was implemented in SIS.

We constructed a set of total five pulse generators as summarized in Table II, where we report pulse width and area. Each pulse generator was designed to drive up to 10 latches with slew constraint of 60 ps, which is enough for safe latching of data. The design considerations of pulse generators will be discussed in Section IV-B.

TABLE I

COMPARISON OF CLOCK PERIOD FROM PULSED LATCH-BASED CIRCUITS OPTIMIZED WITH *PWCS_Optimize*, AND CLOCK PERIOD FROM FLIP-FLOP-BASED CIRCUITS OPTIMIZED WITH 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	547	74	1651	1344	1344	1.00	1792	1502	1642	0.52
s15850	3875	515	1238	955	955	1.00	1238	1092	1129	0.75
s38584	12031	1424	1160	1035	1035	1.00	1294	1161	1178	0.87
b14	6967	245	1761	1400	1400	1.00	1887	1558	1731	0.47
b15	7701	449	1949	1737	1737	1.00	2111	1892	1922	0.86
b21	13959	490	2034	1527	1537	0.98	2182	1682	2014	0.34
i2c	1080	129	964	828	828	1.00	1099	974	1002	0.78
ps2	2004	185	1247	843	964	0.70	1422	1010	1321	0.25
ecc	7117	539	1976	1443	1646	0.62	2153	1615	2012	0.26
usbf	14945	1733	1471	1142	1142	1.00	1554	1299	1424	0.51
Average						0.93				0.56

A. Effectiveness of PWCS_Optimize on Clock Period

The results of *PWCS_Optimize* are shown in columns 4–7 of Table I. The initial clock period after logic synthesis, where we assumed the same pulse width for all latches, is denoted by P_{ini} and is shown in column 4. P_{opt} in column 5 corresponds to optimum clock period, which was obtained after clock skew scheduling on the initial netlist while assuming that arbitrary amount of skew can be assigned. The clock period obtained by *PWCS_Optimize* is denoted by P_{pwcs} and is shown in column 6. The maximum allowable skew (Δ) was assumed to be 10% of P_{opt} [6]; pulse generators (see Table II) with their pulse width within 40% of P_{opt} were allowed for each circuit, so that any path that violates 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 clock period, we introduce its figure of merit (FOM):

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

which is computed for each circuit and is shown in column 7. Note that $0 \leq \eta \leq 1$; the larger η is, the closer P_{pwcs} is to optimum clock period P_{opt}.

Most circuits (except two) achieve $\eta = 1$ or close to 1; two circuits (`ps2` and `ecc`) have rather small η as their P_{opt} is considerably smaller than P_{ini}, thus skew up to 10% of P_{opt} and pulse widths up to 40% of P_{opt} are not enough to yield optimum 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 out that two circuits have many adjacent blocks with large difference of maximum delay, which is why their P_{opt} is considerably smaller than P_{ini} than other circuits. The distribution of pulse generators after running *PWCS_Optimize* is shown in Fig. 4, which demonstrates the numerical domination of PG1, pulse generator of shortest pulse width. If we restrict pulse generators to PG1 alone, the average η becomes 0.45; thus, even though PG1 dominates in numbers, exploiting small numbers of pulse generators with wider pulse width gives us the opportunity of significantly reducing clock period.

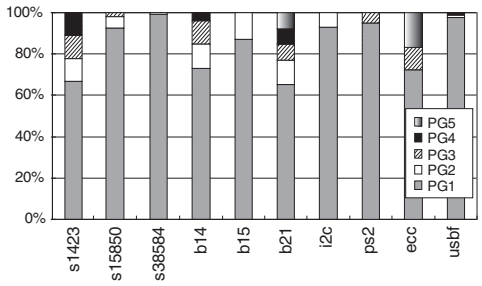

Fig. 4. Distribution of pulse generators after running *PWCS_Optimize*.

We performed similar experiments for flip-flop-based circuits. In the initial netlist synthesized with latches (corresponding to column 4), we substituted flip-flops for all the latches and obtained the initial clock period (column 8). The optimum clock period is shown in column 9, which was obtained via clock skew scheduling [9] while assuming arbitrary amount of skews can be assigned. The clock period with clock skew scheduling while assuming skew of up to 10% of P_{opt} (thus, the same constraint of skew in our approach) is shown in column 10, and its figure of merit is shown in column 11. Comparing optimum clock period of two styles of circuits (columns 5 and 9) demonstrates the benefit of pulsed latch-based circuit due to less sequencing overhead. Comparing P_{pwcs} and P_{css}, with their corresponding η, shows the benefit of pulsed latch-based circuit optimized with *PWCS_Optimize*.

B. Design Considerations of Pulse Generators

A list of available pulse widths $\mathcal{W}$ is important for *PWCS_Optimize* in reducing clock period. We ran *PWCS_Optimize* for the same benchmark circuits in Table I, while we vary the interval of pulse width between consecutive pulse generators (see Table II) from 80 ps to 160 ps (in increment of 10 ps) with PG1 fixed to 131 ps, and obtained the average FOM as shown in Fig. 5. Since most circuits achieve $\eta = 1$ even if we vary the interval of pulse width, the change in average FOM is mainly determined by two circuits (`ps2` and `ecc`), which, in our experiments, give the best average FOM

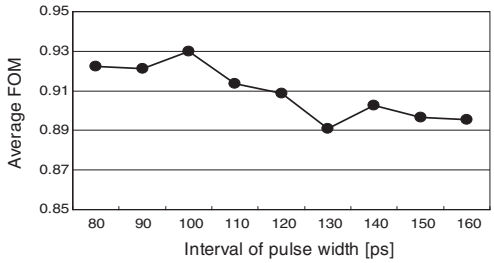

Fig. 5. Average FOM with varying pulse width interval.

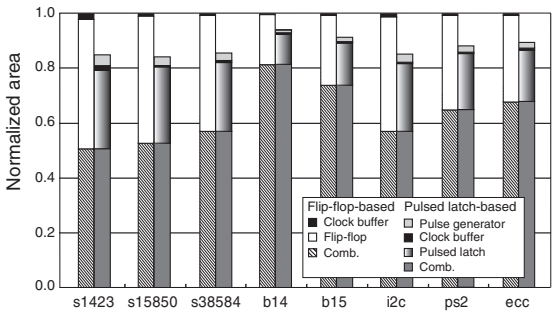

Fig. 6. Comparison of area between flip-flop-based circuits (left bars) and pulsed latch-based circuits (right bars) optimized with *PWCS_Optimize*. All numbers are normalized to total area of flip-flop-based circuits.

for the pulse width interval of 100 ps, as shown in Fig. 5. This is the basis for designing our pulse generators as shown in Table II. More factors, such as uncertainty of pulse width due to process variation, should be taken into account, which is our future work.

C. Physical Design

Fig. 6 compares the area of two styles of circuit: the left-hand bars correspond to flip-flop-based circuits and show the proportions of area from clock buffers, flip-flops, and combinational gates; the right-hand bars correspond to pulsed latch-based circuits after following the design flow shown in Fig. 2, and show the proportions (normalized to total area of flip-flop-based circuit) of pulse generators, clock buffers, latches, and combinational gates. Even though pulsed latch-based circuits involve extra pulse generators, overall area occupied by clocked elements (i.e. elements other than combinational gates) is reduced due to smaller area of latch over flip-flop. The total area of pulsed latch-based circuits are reduced by 12% on average, in which the amount of reduction is determined by the proportion of sequencing elements (thus, reduction is the largest in `s15850` and the smallest in `b14`).

Fig. 7 shows the final layout of an example circuit `i2c`, which was obtained following the design flow in Fig. 2. Pulse generators are marked in red; pulsed latches are marked in green.

V. Conclusion

We have presented a pulsed latch-based design of sequential circuits, focusing on the primary problem of minimizing clock period through pulse width allocation and clock skew scheduling. By using small number of various pulse widths combined with clock skews (up to 10% of clock period), we showed through experiments that we can achieve minimum clock period for many benchmark circuits, and figure of merit of 0.93 on average. The algorithm to find a minimum clock period 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, have been presented and demonstrated. In spite of extra pulse generators, pulsed latch-based circuits have been shown to occupy 12% less area on average compared to flip-flop-based counterpart.

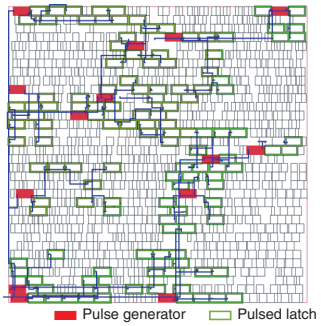

Fig. 7. Layout of `i2c` after allocating pulse generators to latches, placement and routing, and synthesis of clock trees.

Design of pulse generators and local clock tree that are less susceptible to process variations is important; *PWCS_Optimize* considering process variations also needs further investigation.

References

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

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

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

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

[5] S. Shibatani and A. Li, "Pulse-latch approach reduces dynamic power," July 2006, EE Times.

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

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

[8] R.-S. Tsay, "Exact zero skew," in *Proc. Int. Conf. on Computer Aided Design*, Nov. 1991, pp. 336–339.

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

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

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

[12] "Opencores," http://www.opencores.org/.

[13] E. M. Sentovich *et al.*, "SIS: a system for sequential circuit synthesis," May 1992, Tech. Rep. UCB/ERL M92/41.