Pulsed-Latch Aware Placement for Timing-Integrity Optimization

Yi-Lin Chuang, Sangmin Kim, Youngsoo Shin, and Yao-Wen Chang, *Member, IEEE*

***Abstract*—Utilizing pulsed-latches in circuit designs is one emerging solution to timing improvements. Pulsed-latches, driven by a brief clock signal generated from pulse generators, possess superior design parameters over flip-flops. If the pulse generator and pulsed-latches are not placed properly, however, pulse-width degradations at pulsed-latches and thus timing violations might occur. In this paper, we present a unified placement framework for pulsed-latches to maintain the timing integrity. Our new placer has the following distinguished features: 1) a multilevel analytical placement framework to effectively prevent the potential pulse-width distortion problem; 2) a physical-location aware pulse-generator insertion algorithm to identify each desired group of a pulse generator and latches; and 3) a new optimization gradient for global placement to consider the impact of load capacitance of generators. Experimental results show that our placement flow can effectively consider pulse-width integrity and thus achieve much smaller total/worst negative slacks with marginal wirelength overheads, compared to a leading commercial and an academic placement flows.**

***Index Terms*—Physical design, placement, pulsed-latch.**

I. Introduction

FLIP-FLOPS are commonly used as sequential components to store data in datapath circuits. Flip-flop-based sequential circuits enjoy the advantage of easier timing verification since flip-flop synchronization with the edge-triggered clock feature can easily be captured in static timing analysis (to be detailed in Section II-B). As a result, 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 circuits (ASICs). However, since a conventional flop-flop is composed of two latches (master and slave) triggered by a clock signal, flip-flops have significantly more overheads than latches in terms of delay, clock load, and area. In the simulation report under the 45 nm process technology from our experiments, a flip-flop requires 1.25 times of setup time and 1.55 times of area than a latch. Specifically, the delay of a flip-flop is one of many reasons why ASICs are slower than custom designs under the same technology by a factor of six or more [6].

Level-sensitive latch designs are relatively simple and consume much less power than that of flip-flops. However, it is harder to perform timing verification on latch designs due to their data transparent nature. On the other hand, because of the transparency, latches allow a combinational block to have delay larger than the clock period, commonly called time borrowing or cycle stealing. Clock skew can be tolerated if the transparency window, shifted by the skew, can still capture the data [16]. For this reason, they are widely used in high-performance microprocessor designs.

Pulsed-latches are latches driven by a pulse clock waveform. A latch is synchronized with the clock similarly to an edge-triggered flip-flop because the rising and falling edges of the pulse clock are almost identical in terms of timing. In addition to its better design parameters (e.g., area, delay, and so on), therefore, a pulsed-latch itself also offers easier timing verification/optimization just like a flip-flop. In practice, pulsed-latches have been adopted in modern circuit designs [2], [18], [25], [29]. With appropriate pulse widths or retiming for latches, circuit timing can effectively be improved [16], [17], [27], and power consumption can further be improved when clock gating is applied [15]. Therefore, pulsed-latch based designs have become a promising solution for modern circuit designs.

Triggered by a clock signal, pulsed-latches require a pulse generator to generate a clock waveform. Different types of generators generate pulses of different widths. Fig. 1 shows our pulsed-latch design scheme and a pulse generator structure. After receiving the clock waveform from the clock source, a pulse generator with the structure illustrated in Fig. 1(b) generates a brief clock signal to each connected latch. A similar pulse-generator structure is also applied in [16] and [26], and its pulse width is controlled by the delay cell. In this methodology, latches with the same pulse width would share the same generator. In this paper, we call the latches and their corresponding generator as pulse-generator-latches (PGL) group.

Manuscript received March 19, 2011; revised June 19, 2011; accepted August 6, 2011. Date of current version November 18, 2011. This work was supported in part by the National Science Council of Taiwan, under Grant NSC 97-2221-E-002-237-MY3, and by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund), under Grant KRF-2008-331-D00406. A preliminary version of this paper [10] was presented at the 47th Design Automation Conference, Anaheim, CA, June 13–18, 2010. This paper was recommended by Associate Editor I. Markov.

Y.-L. Chuang is with TSMC, Ltd., Hsinchu 300, Taiwan (e-mail: nicky@eda.ee.ntu.edu.tw).

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

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

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.2011.2165717

0278-0070/$26.00 © 2011 IEEE

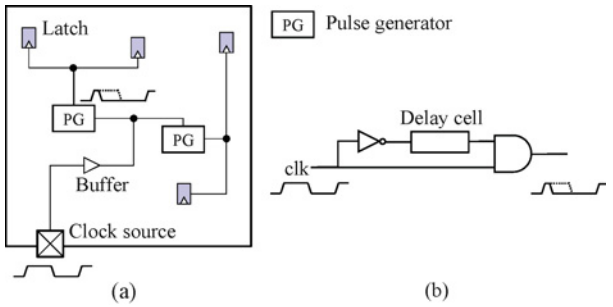

Fig. 1. (a) Pulsed-latch circuit and (b) pulse generator structure, which consists of an inverter, a delay cell, and an AND gate.

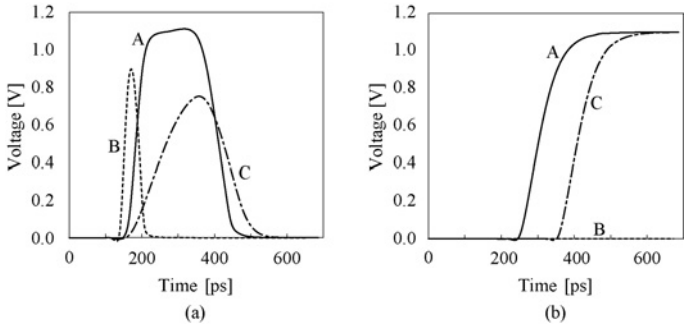

Fig. 2. Pulse shape is distorted as load capacitance increases. (a) Various pulses apply to a latch. (b) Corresponding waveforms of the latch output.

Since a pulse generator itself is a combinational circuit [see Fig. 1(b)], the delay and driving capability of a pulse generator would also be affected by the (output) load capacitance. If a pulse generator and latches are not placed properly, the wirelength among them might become too large and thus distort the generated pulse width, which might cause serious timing violations. Fig. 2 shows the pulse-shape simulation results for different load capacitances. We generated three different clock signals by *A* (0.89 fF), *B* (created by SPICE, which intends to show the result for a narrow signal), and *C* (38 fF). As the load capacitance of a generator keeps increasing, the generated pulse width shrinks dramatically [Fig. 2(a)]. As shown in Fig. 2(b), the input data are not captured under a very narrow pulse *B*, and the output delay for the distorted pulse *C* is much longer than that for the normal pulse *A*. Even if we apply the same width to all latches, the load capacitance of each PGL group also needs to be considered.

In this paper, the timing integrity is evaluated by whether a latch can safely capture its input data and also operate properly. Fig. 3 shows a circuit configuration and experimental results for examining the timing integrity of a pulsed-latch circuit under the 45 nm technology. We changed the number of latches that are driven by a single generator and the value of wire capacitance between a generator and latches. If the data were not captured by any latch, or the clock-to-Q delay of a certain latch exceeded 110% of the nominal value (due to distortion of the pulse shape), we treated it as a failure. See Fig. 3(b) for the corresponding shmoo plot. To further explore the pulse-generator characteristics, we conducted a set of experiments to explore the corresponding effects in Section II-C.

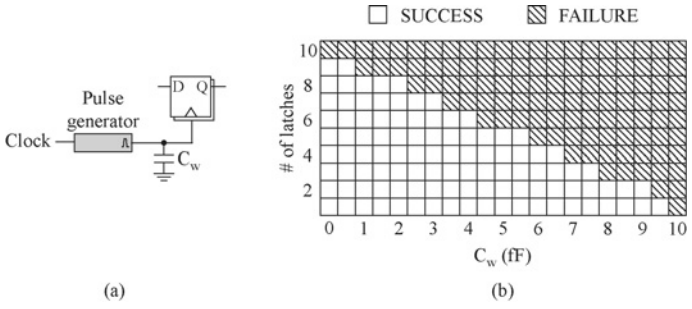

Fig. 3. (a) Circuit configuration to examine the timing integrity. (b) Corresponding experimental results.

Due to the pulse-width degradation property shown in Figs. 2 and 3, the pulsed-latch placement is essentially different from traditional placement problems. In traditional placement problems, the main objective is to minimize the total wirelength; however, minimizing the total wirelength does not mean that the placement result will have smaller load capacitance at a generator. Since traditional placement algorithms do not honor the load capacitance of a generator, this could cause longer wirelength between the generator and latches and thus degrade the pulse width. One simple solution is to increase the net weight between a pulse generator and corresponding latches like traditional clock-gating-aware or power-aware placement. However, it is very difficult to determine the magnitude of increased weight for each PGL group, especially when each PGL group has a different pulse-width profile. Applying an over large weight could maintain the pulse width during placement; however, it could cause unstability problems on wirelength and density. To explore the impact of net-weighting technique on this problem, we also conducted an experiment to compare the results between net weighting and our approach in Section IV-D. As to be shown in the experimental results, applying larger net weights on PGL groups might not maintain pulse width well. Therefore, the traditional placement techniques cannot directly apply to the pulsed-latch placement problem well.

In this paper, we present a unified analytical placement framework to tackle this problem. We summarize the key features of our approach as follows.

1) Our proposed multilevel placement framework keeps latches in local proximity of the corresponding generator during intermediate levels with the load capacitance consideration of each PGL group. Therefore, our proposed placement framework can prevent from timing violations that might occur in the traditional sequential-circuit placement flows.
2) Unlike the previous work [16] that determines PGL groups only by logic specifications from the scheduled netlist, we develop a better PGL-group identification algorithm which can effectively reduce the wirelength overhead by the physical relations among latches.
3) Unlike the simple net-weighting technique used in clock gating or power-aware placement, during the placement, with the proposed optimization gradient, the barrier forces, integrated with the analytical placer, we can effectively consider the interactions among PGL groups

and other logic cells and thus optimize each element simultaneously and globally with better tradeoff between total wirelength and density.

4) Experimental results show that the pulsed-latch based designs placed by our proposed methods result in superior pulse-width integrity. This allows us to obtain significant improvements on timing with comparable wirelength, compared to traditional sequential circuit placement flows.

The rest of this paper is organized as follows. Section II gives a brief review on analytical placement framework, pulse-generator features, and defines our pulsed-latch aware placement problem. Section III details our algorithms. Experimental results are presented in Section IV. Finally, the conclusion is given in Section V.

II. Preliminaries

A. Analytical Placement Framework

Since we adopt NTUplace3 [8] to demonstrate our placement flow, we shall first review its algorithm. For analytical placement, we model a circuit using a hypergraph $H = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denote cell/macro blocks, and hyperedges $E = \{e_1, e_2, \ldots, e_m\}$ represent nets. Let x_i and y_i be the respective x and y coordinates of the center of block i. The analytical placement minimizes wirelength under the cell density constraint, which is modelled using uniform non-overlapping bins. The global placement problem can be viewed as a constrained minimization problem as follows:

$$\begin{aligned} \min \quad & W(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} \quad & D_b(\mathbf{x}, \mathbf{y}) \le M_b \qquad \text{for each bin } b \end{aligned} \tag{1}$$

where $W(\mathbf{x}, \mathbf{y})$ is the wirelength function, $D_b(\mathbf{x}, \mathbf{y})$ is the potential function that represents the area function of movable blocks in bin b, and M_b is the maximum potential representing the total area of movable blocks allowed in the bin. The wirelength $W(\mathbf{x}, \mathbf{y})$ is defined as the total half-perimeter wirelength (HPWL). Since function $W(\mathbf{x}, \mathbf{y})$ is not smooth, it is hard to optimize directly. In our placement framework, we use the log-sum-exp function [21] instead to smooth the wirelength function, and then apply gradient search to find the optimal global placement. The potential function can be computed by the multiplication of the horizontal overlap and the vertical overlap. We use the bell-shaped function [14] to obtain the smoothed potential function $\hat{D}_b(\mathbf{x}, \mathbf{y})$. Nevertheless, the bell-shaped function might generate high "mountains" and deep "valleys" in the chip density map, which makes the gradient search difficult and inefficient in finding a desired solution. As a result, we use the Gaussian function to further smooth the base potential [8].

For NTUplace3, its underlying idea is to evenly spread cells gradually over the whole placement region during global placement. In the NTUplace3 implementation, the inequalities (1) are replaced by equalities $D_b(\mathbf{x}, \mathbf{y}) = M_b$, except in evaluating the density-based termination criterion, and we apply a quadratic-penalty method to optimize the global placement. By doing so, we can explicitly make the block density evenly distributed by penalizing "underfilled" bins as well as overfilled ones, which can facilitate the block spreading process during placement optimization. As a result, we convert the original formulation into a sequence of unconstrained minimization problem of the following form:

$$\min \quad W(\mathbf{x}, \mathbf{y}) + \lambda \textstyle\sum_b (\hat{D}_b(\mathbf{x}, \mathbf{y}) - M_b)^2 \tag{2}$$

by introducing the multiplier λ. As the value of λ increases, the cells gradually spread over the chip until the density constraint for each bin is near-satisfied.

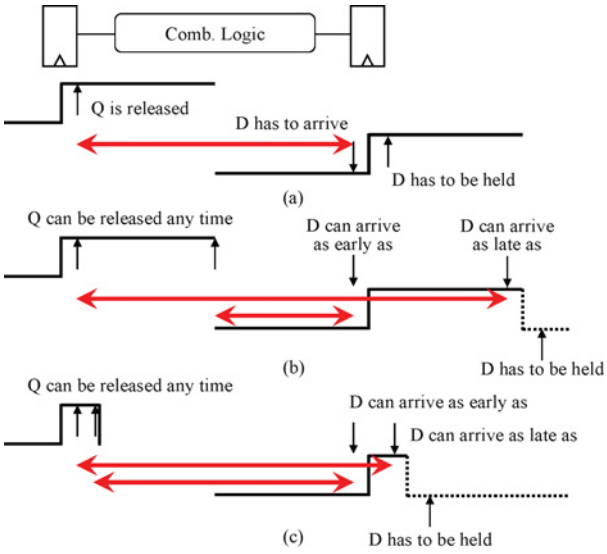

Fig. 4. Timing diagrams of (a) flip-flops, (b) latches, and (c) pulsed-latches. The bidirectional arrows indicate the possible logic computation time.

B. Comparisons Among Sequential Devices

In this subsection, we compare three popular sequential devices and explore the timing advantages of pulsed-latches. Fig. 4 illustrates the timing diagrams between two neighboring sequential devices of flip-flops, latches, and pulsed-latches.

For flip-flops, due to the nature of edge triggering, the device value Q is available after clock-to-Q delay from a rising edge whenever input D arrives; therefore, capturing D and launching Q are independent. This fact simplifies the timing analysis of flip-flop-based designs. For latches, the launch time of Q depends on when D is captured (unless D arrives before the rising edge of a clock signal); therefore, capturing D and launching Q are dependent. As a result, the computation time of the logic might vary for different pairs of latches, which complicates the timing analysis. As observed from the figure, the varied computation times on latch-based designs mainly depend on the duration of a clock signal.

Unlike latches, the duration of a triggering clock signal for a pulsed-latch-based design is typically very short (a brief pulse); as a result, the variation of the computation time can effectively be controlled, which can achieve similar behaviors as flip-flops. In addition, due to the tractable varied computation times between pulsed-latches, we can reduce the resulting clock period by assigning different clock durations to different pairs of latches [16]. Consequently, with better design parameters of smaller area and power consumption

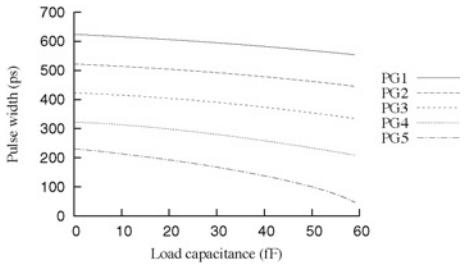

Fig. 5. Relations between load capacitance and pulse width under different types of generators.

as introduced in Section I, pulsed-latches become popular in high-performance circuits.

C. Pulse-Generator Characteristics

As Section I introduces, we consider the load capacitance of generators to reduce the possibility of pulse-width degradation. To understand the impact of capacitance and resultant pulse width, we use HSPICE [12] to find out the relations between these two factors. For each type of generator, we derive the corresponding pulse width under different load capacitances. The relations are summarized in Fig. 5; it shows the capacitance-width relations derived from five pulse generators in this paper. From the figure, we can see that, as load capacitance increases, the resultant width keeps decreasing. If the pulse width is too small, the latch allocated by this width may not function in a proper time, which might increase the clock period (slower design) and cause timing violations.

D. Problem Formulation

In this paper, we try to avoid timing violations incurred by pulse-width degradation of generators. Given a user-specified maximum pulse-width bound for each type of generator, we compute the corresponding maximum tolerable load capacitance from the simulation shown in Fig. 5. To consider the capacitance impact, we transform it into a wirelength constraint [the maximum accumulated wirelength constraint (MAWC)] in the placement stage by our proposed algorithm. Accompanied by a scheduled netlist, we can determine a set of PGL groups with logical consideration and physical correlations. We formulate our problems as follows:

1) *Pulsed-Latch-Aware Placement:* Given a pulsed-latch-width scheduled netlist with the maximum tolerable load capacitance of each type of generator: 1) determine PGL groups, and 2) find the corresponding placement of blocks such that the total wirelength is minimized and specified maximum tolerable capacitance is satisfied.

III. Multilevel Pulsed-Latch-Aware Placement Framework

In this paper, we propose a multilevel analytical placement framework for pulsed-latch designs. Fig. 6 summarizes the flow. Before placement, since the netlist does not contain generators yet, we first determine a set of PGL groups by the physical locations from an initial placement. After this process, we update the input netlist with generators connected to each

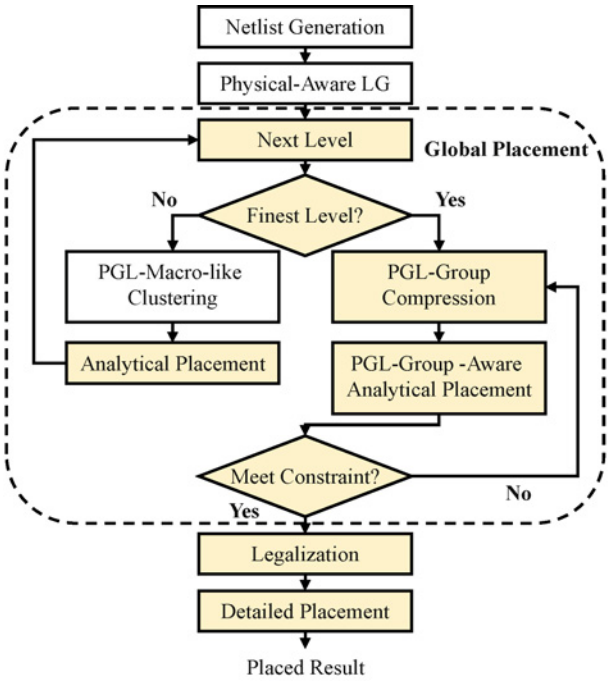

Fig. 6. Proposed flow of our multilevel analytical placement framework for pulsed-latch designs.

set of latches. Due to increasing complexity of modern circuit designs, multilevel framework [5], [8], [20], [31] is usually applied on analytical placement to improve the scalability. In general, the generator and latches of each PGL group should be placed closer to each other due to the maximum tolerable capacitance constraint. We propose a PGL-macro-like clustering technique to provide close distribution of latches in each PGL group before solving the analytical placement formulation. Then in the finest level, with known block distributions, we apply PGL-group aware global placement techniques including PGL-group compression and the proposed barrier force. In this level, our proposed optimization scheme provides a correlation mechanism among the traditional wire force, the density spreading force, and the barrier force to derive a global placement result with the MAWC consideration. Finally, after the legalization and detailed placement, we can obtain a placement result of pulsed-latches. In the following subsections, we will detail the proposed algorithms.

A. Physical-Aware Latch Grouping

Given a pulse-width scheduled netlist and a timing-driven initial placement, we first determine each PGL group by the physical locations of latches. In [16], Lee *et al.* determined the PGL groups only by the logic relations and skew specifications, which may not be sufficient for pulsed-latch placement. To obtain a better tradeoff between wirelength and the maximum capacitance constraint, we resort to a new grouping method with physical information in a circuit placement. We present our physical-aware latch grouping (PLG) algorithm in Fig. 7.

There are two major issues in the PLG algorithm: PGL-group and MAWC specification. The PLG algorithm checks each type of pulse-width latch and identifies an appropriate

Algorithm: **Physical-Aware Latch Grouping**
Input:
H: a pulse-width scheduled netlist by [16]
C_t: maximum tolerable capacitance of generator type t
l_i: a set of latches
Output:
H': updated netlist including pulse generators connected to the corresponding set of latches with the same pulse width

01. **for** each pulse-width type $t \in H$
02. un-group latches $U = \{l_i\}$
03. **while** $U \neq \phi$
04. find the starting point l in the most congested region
05. $G = \{l\}, U = U - \{l\}$
06. **while** $total\_capacitance(G) \leq C_t$
07. find a latch $l_i \in t$ with minimum $distance(l_i, l)$
08. $G = G + \{l_i\}$ and $U = U - \{l_i\}$
09. generate a group G
10. update latch density regions
11. generate a set of groups G_i with type t
12. generate all sets of groups G_i with all types t
13. insert corresponding generators
14. update the netlist H to H'
15. output H'

Fig. 7. Our PLG algorithm.

group by estimating the potential accumulated capacitance, without exceeding the maximum capacitance constraint of the corresponding type of generators. Based on an initial placement, we transform the capacitance into the impact of the MAWC within each PGL group. After this transformation, satisfying the MAWC means considering the maximum capacitance constraint implicitly.

Each group size is determined by the accumulated capacitance. For each grouping iteration, we consider the following inequality (line 6 in Fig. 7):

$$C_w + C_l \leq C_t \tag{3}$$

where C_w and C_l are the capacitance of wire and latches, respectively. C_t is the maximum tolerable capacitance of the generator with pulse width t. C_w can be obtained by the unit capacitance of metal wire and the corresponding wirelength, which in turn can be expressed by $C_w = W_l(\mathbf{x}, \mathbf{y}) \times C_u$, where $W_l(\mathbf{x}, \mathbf{y})$ and C_u are the wirelength function of a latch group and unit capacitance of a metal wire, respectively. In this process, we use the HPWL to estimate the interconnect length among latches. We continue to group latches if (3) is satisfied.

After all latches are grouped, we have the grouping result. As a result, the MAWC for each PGL group is set as the value of $W_l(\mathbf{x}, \mathbf{y})$. Then we update the netlist by adding the generators and connect the generator to a set of designated latches with the same pulse width. Based on the proposed grouping algorithm, we can effectively reduce the wirelength and fully utilize the pulse generators, which provides a much better PGL-grouping result compared with the previous work.

B. PGL-Macro-Like Clustering

The multilevel framework adopts a two-stage technique of bottom-up coarsening followed by top-down uncoarsening. During the coarsening stage, the blocks are clustered level by level to reduce the number of movable blocks. The clustering process continues until the number of blocks is reduced below some threshold. After clustering, the analytical placement problem is solved at each level of the uncoarsening stage. However, as we mentioned in Section II-D, we intend to derive a placement satisfying the MAWC. If the multilevel framework is not aware of this requirement before solving the analytical formulation, after declustering, the blocks of a PGL group may spread all over the placement region, which makes the analytical placer difficult to meet the MAWC.

Moreover, since in the uncoarsening stage, each cluster contains multiple movable blocks, and the exact placement within a cluster remains unknown, it is relatively difficult to optimize PGL-group locations within each cluster directly. Therefore, in the upper level of uncoarsening, we cluster the blocks of each PGL group to a single macro and update the corresponding nets connecting to the original latches or generators. Then in the multilevel framework, each PGL group can be regarded as a small macro in the placement region. As the analytical placer gradually optimizes the clustered blocks level by level, the placer implicitly provides a better location for each of PGL groups.

In addition, to effectively consider the cluster size, we adopt the best-choice clustering [1] to obtain the clusters during each level. Therefore, in the finest level after each PGL macro is declustered, the latches belonging to the same PGL group would have relatively closer distances, which provides a good initial input for analytical placer.

C. PGL-Group-Aware Placement

1) *Barrier Method for Global Placement of Pulsed-Latches:* When the uncoarsening stage reaches the finest level, the optimization process of analytical placement framework is directly applied on blocks of the circuit instead of clusters. Therefore, with known latch positions, we can optimize the MAWC in global placement. To force each PGL group to satisfy the MAWC and to maintain wirelength quality, in the finest level, we propose a new barrier force to provide the third gradient for the optimization.

Before we introduce our proposed barrier force, we shall briefly introduce how the barrier method works. Consider the following constrained minimization problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & a_i^T x \leq b_i \quad i = 1, \ldots, m. \end{aligned} \tag{4}$$

To solve this constrained optimization problem, one popular approach is the barrier method. By defining the logarithmic barrier, we solve a sequence of unconstrained minimization problems as follows:

$$\min c^T x + \sum_{i=1}^{m} -\lambda_r \log(-(a_i^T x - b_i)) \tag{5}$$

where λ_r denotes the parameter of the barrier penalty and the logarithmic barrier $\phi(x)$ is defined as [3]

$$\phi(x) = \begin{cases} \sum_{i=1}^{m} -\log(-(a_i^T x - b_i)), & Ax < b \\ +\infty, & \text{otherwise.} \end{cases} \tag{6}$$

With the above definition, we can find that $\phi(x)$ will approach ∞ as x approaches the boundary of $Q = \{x|Ax < b\}$, which can effectively force the solution to lie inside the constraint. Since we want to ensure that the wirelength of a PGL group does not exceed the corresponding MAWC, this property is very useful for this problem. In addition, according to the log-barrier method, if we start at a feasible initial solution x_0 (x_0 satisfies the constraint), we can always stay strictly in the interior of the constraint space [3]. This feasibility property is desirable. Therefore, by adopting the log-barrier method, we have the following two theorems, which can be found in most nonlinear optimization texts. For better readability, we provide the corresponding descriptions/proofs as follows.

Theorem 1 (Feasibility of the Log-Barrier Method): By the log-barrier method, starting at a feasible x_0 in the interior of Q, we can always stay strictly in the interior of Q.

Proof: Consider the general form shown in (4), and assume that the initial solution x_0 is a feasible solution, i.e., $Ax_0 < b$. As illustrated in (6), if x_0 stays in feasible regions, the objective is defined as $c^T x_0 + \sum_{i=1}^{m} -\log(-(a_i^T x_0 - b_i))$; otherwise the objective is infinite. To minimize the objective function, the next solution x' that leads to a non-increasing objective value is always chosen. Therefore, we can guarantee that x' always remains feasible once the initial solution is feasible. ■

Theorem 2 (Smoothness and Convexity Properties of the Barrier Function): According to the first-order and second-order derivatives, the barrier function $\phi(x)$ is a smooth function on Q, and the function is convex for all $y \in \mathbf{R}^n$.

Proof: Considering the form shown in (6), let $d = (1/(b_1 - a_1^T x), \ldots, 1/(b_m - a_m^T x))$. The corresponding first-order derivative of the barrier function is given by

$$\phi'(x) = \sum \frac{a_i}{b_i - a_i^T x} = A^T d \tag{7}$$

where the denominator $b_i - a_i^T x$ is always a non-zero value if x is in Q, implying that there are no non-differentiable points in the barrier function. According to the function $A^T d$, as a result, the barrier function is a smooth function in Q. To show the convexity property of the barrier function, we derive the second-order derivative as follows:

$$\phi''(x) = \sum \frac{a_i a_i^T}{(b_i - a_i^T x)^2} = A^T \mathrm{diag}(d)^2 A \tag{8}$$

where $\mathrm{diag}(d)$ denotes the diagonal matrix of d. For all $y \in \mathbf{R}^n$, we have

$$y^T \phi''(x) y = y^T A \ \mathrm{diag}(d)^2 A y = \|\mathrm{diag}(d) A y\|^2 \geq 0. \tag{9}$$

From (9), since the second-order derivative of the barrier function is nonzero, the barrier function also satisfies the convexity property in Q. ■

To effectively consider the MAWC of each PGL group, we apply the log-barrier method to optimize the placement of pulsed-latches. For the MAWC of each PGL group, the corresponding Q can be defined as $W_g(\mathbf{x}, \mathbf{y}) \leq MAWC_g$,

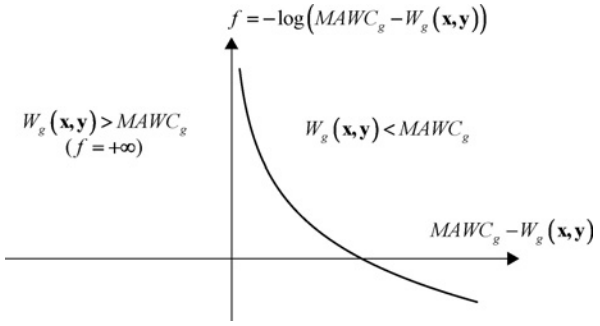

Fig. 8. Definition of the log-barrier formulation for the constraint of accumulated wirelength.

where $W_g(\mathbf{x}, \mathbf{y})$ is the wirelength function of a PGL group g, and $MAWC_g$ is the corresponding MAWC value of the group. Clearly, according to the transformation shown in (5), we can transform our placement problem into an unconstrained formulation as

$$\begin{aligned} \min \quad & W(\mathbf{x}, \mathbf{y}) + \lambda_d \sum_b (\hat{D}_b(\mathbf{x}, \mathbf{y}) - M_b)^2 \\ & -\lambda_r \sum_g \log(MAWC_g - W_g(\mathbf{x}, \mathbf{y})) \end{aligned} \tag{10}$$

where λ_d and λ_r are density and barrier weights, respectively. Fig. 8 depicts the curve of the proposed log-barrier formulation for the wirelength constraint. We observe that the cost of the log term diverges to positive infinity as the PGL-group wirelength $W_g(\mathbf{x}, \mathbf{y})$ approaches its upper bound $MAWC_g$.

As λ_r gradually decreases, the placement result can eventually converge to a desired one with a good tradeoff between wirelength and block density while MAWC is satisfied. For $W_g(\mathbf{x}, \mathbf{y})$, we use the log-sum-exp wire model. To solve this unconstrained form, we apply the conjugate gradient method. By Theorem III-C1, the log barrier in (10) is also a smooth and differentiable function; consequently, we directly compute its gradient in the analytical placement framework.

2) *Feasible Starting Point and Analytical Interpretation:* To apply the log-barrier method, we need to start at a feasible initial solution, implying that the wirelength of each PGL group does not exceed its MAWC. Nevertheless, this requirement might not be preserved after circuit declustering. To alleviate this potential problem, we present the PGL-group compression. We first derive the corresponding vectors from a generator to a latch, and directly move the latch closer to the generator according to the derived vector. We perform the compression by the following equation:

$$X' = \mathrm{diag}(1 - \alpha_i) X + x_g \tag{11}$$

where X is the original x-coordinate matrix of latches, *diag* is the diagonal matrix with diagonal entries $(1 - \alpha_i)$, and X' is the resultant x-coordinate; α_i is a user specified parameter to control how much to compress the latch i along the vector direction, and x_g is the x-coordinate of generator. A similar equation applies to the y-direction.

By applying the barrier force, as shown in Fig. 9, each PGL group receives three different forces simultaneously. Imagine that we have a MAWC feasible region (of course, it is hard

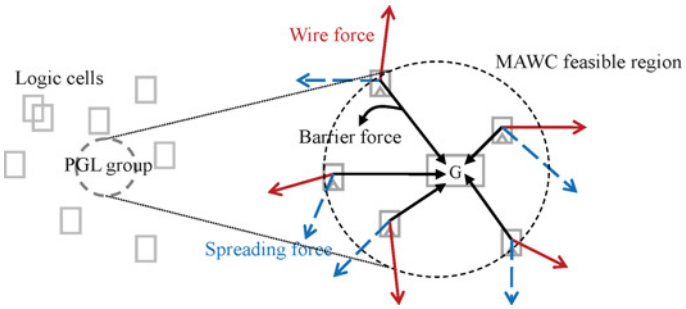

Fig. 9. Force concept of barrier forces among other two traditional forces.

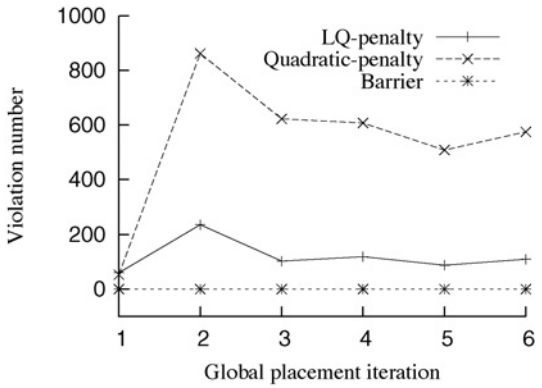

Fig. 10. Violation comparison among our proposed scheme (the log-barrier method), quadratic-penalty, and LQ-penalty approaches.

to find this region exactly in practice), the latch locations are determined by the summation of wire forces, spreading forces, and barrier forces. The barrier gradient provides a confined force that tries to keep the current MAWC region feasible; the latch locations will be guided by the wire and spreading forces. By the proposed optimization scheme, the locations of latches and the generator in a PGL group are determined while considering the impacts of other logic cells simultaneously, which can have higher flexibility to obtain a better placement result.

Conceptually, as the global placement process proceeds, the underlying idea of our proposed analytical formulation can be interpreted as resource-demand interactions. In our global placement, the barrier weight would gradually decrease, the wire and spreading forces are the relatively dominant ones. The value of the MAWC is the total available resources, and the wire and spreading forces would continuously request the resources to minimize their costs (wirelength and density overflow) for shorter wirelength and more even distribution. Therefore, the optimized placement result means that the resources of the MAWC are fully utilized, which can obtain the best tradeoff among these three considerations.

3) *Comparisons Among Different Penalty Methods:* As we introduced in Section II-A, we use the quadratic-penalty method to relax the density constraint into the objective function. The question is whether we can use the same quadratic-penalty for the placement of pulsed-latches. To address the essential difference between the quadratic-penalty and log-barrier methods, we implemented the quadratic-penalty method to consider the MAWC. In addition, to consider different penalty methods, we also implemented a combined penalty of linear and quadratic terms. The detailed formulation and results are summarized as follows.

For the quadratic-penalty method of the MAWC, we can implement $\sum(MAWC_g - W_g(\mathbf{x}, \mathbf{y}))^2$, instead of the original log-barrier form. Another possible penalty function could be a combination of linear and quadratic penalty (LQ-penalty). In this problem, we formulate the LQ-penalty by

$$LQ = \left(\max_g \{MAWC_g - W_g(\mathbf{x}, \mathbf{y})\}^2 \right). \tag{12}$$

As (12) shows, the underlying idea is similar to the quadratic-penalty method. We impose a larger cost on a pulsed-latch if the wirelength between the pulsed-latch and a pulse generator is larger. However, unlike the quadratic penalty, we use the function max to quantify different penalty forces on different PGL groups. When computing cell gradients, we use the log-sum-exp method to smooth the function max. To compare the effects of different methods, we study the number of violations during global placement and also compare the final wirelength and total negative slack (TNS). The violation number is defined as the total number of PGL groups violating the MAWC. In summary, applying the quadratic-penalty method would incur almost 10% increase on the total wirelength and three times worse on TNS. Compared with the barrier method, the total wirelength overhead of the LQ-penalty is 1% and its TNS is 2.47 times worse.

To see the effects of these methods, we explore the violation numbers during global placement. Fig. 10 depicts the violation comparisons of the circuit des_perf for the OpenCores [22] benchmark during its global placement in the finest level. As expected, the log-barrier method can satisfy the MAWC constraint for each PGL group. In contrast, the quadratic-penalty and LQ-penalty methods incur violations. These violations may make the wirelength of a PGL group longer, incurring a higher capacitance load for the generators. Thus, more MAWC violations also imply more timing violations.

The reason for large number of violations in the quadratic-penalty method is that since the method is just a penalty function, it does not need to be satisfied all the time. In other words, if the placer finds that it will have larger gain by making wirelength shorter or overflow smaller, the MAWC will not be conserved. It is clear that the LQ-penalty method can achieve fewer violations than the quadratic-penalty one. The reason is that for a PGL group with an excessive load, the smoothed function max of the LQ-penalty method will impose much larger forces than that of the quadratic-penalty one. However, both methods still generate significant numbers of violations, revealing their inferior effectiveness in tackling the addressed problem.

D. Pulsed-Latch-Aware Analytical Placement Algorithm

For the analytical placement introduced in Section II-A, a multilevel framework is applied to improve the scalability. The hierarchy of clusters is built by a clustering algorithm in the coarsening stage. Then the placement problem is solved

Algorithm: **Pulsed-Latch Aware Placement Algorithm**
Input:
H_0: a hypergraph as the placement result after declustering from level 1
Output:
(x^*, y^*): optimal block positions (including latches/generators)

01. initialize bin grid size n_b
02. initialize base potential for each bin
03. initialize $\lambda_w = 1$
04. initialize $\lambda_d = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$
05. $\gamma = Get\_MAWC\_Factor()$
06. initialize $\lambda_g = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\gamma \sum |\partial \hat{B}_g(\mathbf{x},\mathbf{y})|}$
07. $PGL\_Compression()$
08. **do**
09. solve min $\lambda_w W(\mathbf{x},\mathbf{y}) + \lambda_d \sum_b (\hat{D}_b(\mathbf{x},\mathbf{y}) - M_b)^2 - \lambda_g \sum_g \log(MAWC_g - W_g(\mathbf{x},\mathbf{y}))$
10. $\lambda_d = \lambda_d \times r_d$
11. $\lambda_g = \lambda_g / Get\_MAWC\_Factor()$
12. **if** ($overflow\_ratio < 10\%$)
13. call $LookAheadLegalization()$ and save the best result
14. compute $overflow\_ratio$
15. **until** (spreading enough or no further reduction in $overflow\_ratio$)
16. update block positions

Fig. 11. Our global placement algorithm of pulsed-latches.

from the coarsest level to the finest level. The placement for the current level provides an initial placement for the next level. When blocks are clustered, the interconnections among blocks within the same cluster are ignored. Therefore, for the levels above the finest level, the wirelength estimation of each PGL group cannot be determined exactly, and thus the barrier-method cannot be applied directly on the latches. As a result, we shall apply our placement algorithm on the finest level of the analytical framework.

Our global placement algorithm is summarized in Fig. 11. As mentioned above, the algorithm takes the finest-level hypergraph (after declustering from level 1) as an input. Lines 1–6 are the initialization steps. For each block within a PGL group, its location would be affected by three types of gradients: the wirelength force, the density spreading force, and the barrier force. In order to obtain the best balance among these three forces, we initially normalize wirelength and density forces to the same magnitude, and make the barrier force smaller to facilitate the solution convergence. We detail these ideas as follows. To solve (10) efficiently, we gradually decrease the weight of the barrier force. However, the barrier weight λ_g cannot be over-reduced since we have to maintain the feasible solution (i.e., satisfy MAWC). In addition, due to the ill-conditioning of the log term near the constraint boundary, λ_g should not be reduced too quickly. The barrier force will not work properly if the wirelength of a PGL group increases too fast (and then goes beyond the MAWC). Thus, it is necessary to keep PGL groups strictly feasible when computing cell gradients in placement iterations. Considering this fact and being inspired by the work [23], we propose the smoothed MAWC weighting function *Get_MAWC_Factor* to compute the weighting factor γ. γ effectively considers the ratio of the current wirelength of PGL groups and the total MAWC values

TABLE I
STATISTICS OF THE OPENCORES CIRCUITS AND CLOCK PERIODS AFTER PULSE-WIDTH SCHEDULING

Circuit	#Latches	#Gates	#Nets	P_{pwcs} (ps)	Util (%)
ac97_ctrl	2199	10 205	10 371	439	74.83
aes_core	530	15 796	16 055	1208	69.21
des_perf	8808	74 011	74 224	878	72.90
mem_ctrl	1045	11 672	11 782	1552	72.28
pci_bridge32	3134	20 248	20 743	1006	74.29
s38417	1463	8875	8904	1488	75.06
systemcases	670	9241	9500	2000	72.43
wb_conmax	770	30 629	31 721	924	70.72

σ to help the analytical placer provide appropriate barrier forces, which can be expressed as

$$\gamma = 1 + \frac{h}{1+e^{k(0.5-\sigma)}} \tag{13}$$

where $\sigma = \sum W_g(\mathbf{x},\mathbf{y}) / \sum MAWC_g$ ($\sigma \leq 1$) and h, k are user-specified parameters. We take the wirelength gradient as a reference for the initial weight of other two forces. As a result, we set $\lambda_w = 1$, $\lambda_d = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\sum |\partial \hat{D}_b(\mathbf{x},\mathbf{y})|}$, and $\lambda_g = \frac{\sum |\partial W(\mathbf{x},\mathbf{y})|}{\gamma \sum |\partial \hat{B}_g(\mathbf{x},\mathbf{y})|}$, where $\partial \hat{B}_g(\mathbf{x},\mathbf{y})$ is the magnitude of smoothed barrier forces.

Lines 8–15 are the main loop to solve the global placement problem for pulsed-latches. To achieve a placement result with fewer block overlaps, we iteratively increase the density weight λ_d by r_w, and decrease the barrier weight λ_g by γ, as explained above. Since it is not easy to determine when the blocks are spread enough for the later legalization stage, we try to legalize the global placement result during global placement to ensure that the current global placement result is good enough. Thus, when the blocks spread enough, we apply look-ahead legalization (*LookAheadLegalization*) [8], [11] to legalize cells and record the legalized wirelength. For each iteration with small block density overflows, we compare the wirelength of the current legalized placement with the shortest wirelength found in past iterations, and then we save the placement with the shorter wirelength. If the number of inferior solutions (larger wirelength) exceeds a threshold, look-ahead legalization will be terminated.

IV. EXPERIMENTAL RESULTS

We conducted experiments to verify the effectiveness of our proposed flow and placement algorithm. Our placement algorithm was integrated into NTUplace3 [8]. It should be noted that the proposed algorithm is flexible and can also easily be integrated into other placers based on the similar framework. We compared the proposed placement algorithm with two other placers, Cadence SoC Encounter [4] (a leading commercial tool) and DCTB [28] (a state-of-the-art academic clock-tree aware placer with its name taken from the paper title). To further explore the impact of latch grouping, we analyzed the tradeoff between the wirelength increase and the size of a PGL group. In addition, we conducted experiments to compare our method with the net-weighting approach mentioned in Section I and the PLG algorithm with PGL-group refinement.

Most experiments were performed on the same machine with four AMD Opteron CPUs and 16 GB memory. We

TABLE II
COMPARISONS AMONG THE THREE PLACEMENT FLOWS

Circuit	Flow A: LC + Encounter									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	18	63	5.54	27.88	16.80	152.35	31.49	12.94	2.36	15.30
aes_core	133	53	27.41	48.36	91.99	1203.10	129.54	32.53	1.56	34.09
des_perf	56	387	15.98	100.00	50.12	586.70	1030.04	132.28	15.53	147.81
mem_ctrl	300	93	16.11	51.65	71.98	1034.41	122.48	18.08	2.14	20.22
pci_bridge32	289	274	18.71	66.17	122.93	3776.37	455.31	31.72	7.68	39.40
s38417	50	64	7.34	42.08	2.48	60.85	57.27	9.70	1.42	11.12
systemcases	3	18	15.39	29.77	54.15	67.64	3.46	22.35	1.78	24.13
wb_conmax	743	52	21.17	100.00	202.10	8313.09	130.27	109.14	3.37	112.51
Norm.	–	–	17.63	11.00	(11.07)	(40.58)	–	0.99	5.57	1.07
Circuit	Flow B: LC + DCTB									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	35	74	5.56	29.49	12.51	170.13	48.40	12.79	2.29	15.08
aes_core	81	53	24.04	62.31	56.47	539.57	130.36	32.61	1.40	34.02
des_perf	72	449	15.18	100.00	138.85	708.05	1167.60	128.37	15.87	144.24
mem_ctrl	155	92	15.35	39.77	45.97	318.52	135.35	18.80	2.06	20.86
pci_bridge32	147	273	16.40	100.00	166.85	1641.01	456.19	29.41	7.06	36.47
s38417	80	61	6.51	28.84	5.54	177.66	47.15	9.62	1.28	10.91
systemcases	43	55	13.71	51.02	51.18	828.61	70.79	18.70	1.56	20.26
wb_conmax	722	51	18.81	100.00	190.84	8000.48	143.29	124.37	1.75	126.12
Norm.	–	–	16.03	12.62	(12.08)	(24.40)	–	0.97	4.57	1.05
Circuit	Flow C: Our flow									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	10	0	0.86	5.87	1.95	8.04	0	13.56	0.92	14.48
aes_core	7	0	1.10	5.39	5.12	13.20	0	31.91	0.23	32.15
des_perf	13	0	0.69	7.66	5.57	17.45	0	132.92	4.94	137.86
mem_ctrl	5	0	0.80	5.60	3.62	10.01	0	19.01	0.50	19.51
pci_bridge32	74	0	0.79	3.65	8.41	382.89	0	29.35	1.33	30.68
s38417	27	0	1.02	5.21	1.21	19.93	0	11.09	0.47	11.56
systemcases	0	0	0.84	3.88	0	0	0	19.92	0.23	20.15
wb_conmax	29	0	1.23	5.53	37.63	345.10	0	122.83	0.30	123.13
Norm.	–	–	1.00	1.00	1.00	1.00	–	1.00	1.00	1.00

Note that flow C is our proposed flow, which is PLG + pulsed-latch aware placer. The values in row "Norm." are all normalized to those of flow C, and normalized WNS/TNS values in flows A and B are in parentheses due to the denominator "0" in our flow.

tested on the eight OpenCores [22] circuits in the IWLS2005 benchmark suite [13]. The circuit statistics are given in Table I. The columns "#Latches," "#Gates," "#Nets," "P_{pwcs}," and "Util (%)" list the number of latches, the number of gates, the number of nets, the clock period reported by [16], and the utilization rate of the circuit, respectively. Our comparisons for the timing results are based on the clock period P_{pwcs}.

We synthesized the circuits and added pulse generators using the Nangate 45 nm Open Cell Library [19]. A set of five pulse generators were constructed to provide pulse widths, including 230 ps, 322 ps, 423 ps, 522 ps, and 623 ps. Each generator has the similar structure and layout with [16] and [26]. In this paper, the pulse width is controlled by the delay cell.

A. Comparisons Among Design Flows

In this experiment, we compared three different flows: (A) LC (latch-clustering) followed by SoC Encounter; (B) LC followed by DCTB; and (C) our proposed flow introduced in Section III. The input netlists used in flows A and B were generated by the LC method proposed in [16]. For flow C, we used the PLG proposed in this paper to determine the PGL groups. Each generated netlist with generators is fed to the three placers: Encounter with timing-driven mode, DCTB, and our pulsed-latch aware placer. To compare with the flow integrated with DCTB, we also implemented the DCTB algorithm into NTUplace3 for fair comparison.

After circuit placement, we used FLUTE [7] to estimate the clock Steiner wirelength and derived the corresponding wire capacitance. Using the simulation results shown in Fig. 5, we calculated the corresponding pulse width by interpolation, and applied the derived pulse widths to latches for timing verification. Given a clock period, we explored the internal impact of pulse-width degradation after circuit placement. To quantify this measure, we propose the average pulse-width degradation ratio (APDR) and maximum pulse-width degradation ratio (MPDR) to model the width degradation of each latch. Given the allocated pulse width $P_{i,\text{opt}}$ of latch i in a scheduled netlist (which does not consider any physical information and can thus be regarded as the optimal period), the APDR is defined as

$$APDR(P_{\text{opt}}, P_{\text{placed}}) = \frac{\sum_{i \in L} \left(\frac{P_{i,\text{opt}} - P_{i,\text{placed}}}{P_{i,\text{opt}}} \right)}{|N_L|} \tag{14}$$

where L is the set of latches, N_L is the total number of latches, and $P_{i,\text{placed}}$ is the pulse width of latch i after placement. The MPDR is defined as

$$MPDR(P_{\text{opt}}, P_{\text{placed}}) = \max_{i \in L}\{\frac{P_{i,\text{opt}} - P_{i,\text{placed}}}{P_{i,\text{opt}}}\}. \quad (15)$$

APDR and MPDR can be considered as the timing indices of pulse widths. A smaller APDR implies that the placement result may maintain the scheduled pulse widths more easily and is thus more likely to have a smaller clock period. Moreover, in addition to the degradation ratio, we also report the total excessive wirelength ratio (EWLR) of PGL groups in terms of corresponding MAWCs. The EWLR evaluates how much the resulting placement violates MAWC, and the EWLR is defined as

$$EWLR = \sum_{g} \frac{\max\{\left(W_g(\mathbf{x}, \mathbf{y}) - MAWC_g\right), 0\}}{MAWC_g}. \quad (16)$$

The experimental results for the three different flows are shown in Table II. The columns "TPV," "LGV," "WNS," "TNS," "SW," "CW," and "TW" are the total number of timing-path violations, the total number of PGL groups violating MAWC, the worst negative slack, the total negative slack, the signal wirelength, the clock wirelength, and the total wirelength, respectively. The WNS and TNS were computed from SIS [24] by setting the clock period to the original optimal clock period, and adding all the slacks of the timing paths with violations.

As Table II shows, compared with flow A, our proposed flow achieves 18, 11, and 41 times improvement on APDR, WNS, and TNS, respectively. Compared with flow B, we achieve 16, 12, and 24 times improvement on APDR, WNS, and TNS. The MPDRs of flows A and B are much larger than that of our flow, revealing excessive loads in certain generators with these two flows (100.00 in MPDR means that the pulse width drops to zero due to an excessive load). Moreover, our flow achieves the minimum violations. Observed from EWLR, further, other PGL groups in flows A and B violate MAWC significantly. The wirelength violations correspond to timing (WNS/TNS) violations as well. The reason for these substantial improvements consists of two factors. First, the LC algorithm does not consider physical relations among latches; therefore, the quality of its placement result is limited. Besides, the placement algorithms of SoC Encounter and DCTB do not take account of the characteristics of pulse generators, which leads to inferior performance for pulsed-latch placement.

With our barrier method, we reduce the clock wirelength by almost five times, compared with other flows. With the effective interaction among wire forces, in addition, our respective signal wirelength overheads over flows A and B are 1% and 3%. We summarize the runtime and area as follows. Since our PLG algorithm considers load capacitance based on the physical correlation among latches, we need more generators than the LC method, ranging from 21 to 397 generators which sum up to a 3.40% area overhead than flows A and B on average. In addition, according to the reported runtime, flows A and B each incur 55% and 61% longer runtimes than our flow. Note that the commercial tool (flow A) is installed on an Intel Xeon 64-bit machine with 8 GB memory, so the reported runtime is for the purpose of reference. For flow B, DCTB would impose additional attracting forces to the pairs of topological clock tree nodes during global placement, which could cause solution oscillation during global placement. Therefore, it might incur redundant loops and thus degrade the solution quality and efficiency.

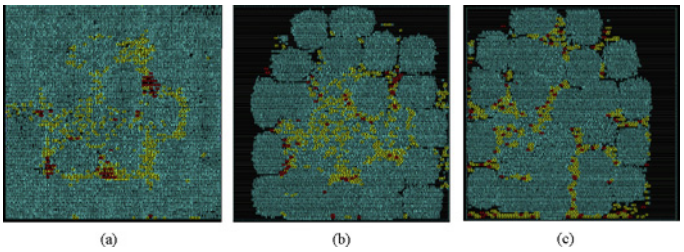

Fig. 12. Snapshot of the placement of circuit aes_core derived from the three placement flows. (a) LC+Encounter. (b) LC+DCTB. (c) Ours. The red nodes are generators, and the yellow ones are latches.

Fig. 12 shows the placement snapshot of circuit aes_core for the three placement flows. It can be observed that the distances among generators and latches in Fig. 12(a) and (b) are much larger than ours (flow C). Therefore, flows A and B are more susceptible to distortions of pulse shapes, which is likely to cause timing violations.

B. Comparisons Among Placement Algorithms

In this experiment, we compared individual placement algorithms. The input netlist consists of two different PGL-group configurations, derived from LC and PLG. We then applied Encounter, DCTB, and our pulsed-latch aware placer to place the circuits. The experimental results are shown in Tables III and IV. Note that the detailed values of Encounter (DCTB) under LC and our approach (PLG+our placer) can be found in Table II.

From the reported results, we can see that our PLG algorithm identifies better PGL groups than the previous works. The placers embedded with the PLG algorithm can significantly reduce PGL-group violations (LGV) and EWLR, justifying that the proposed latch-clustering scheme is effective for pulsed-latch placement. For the LC scheme, our placement algorithm achieves about three times, two to three times, and four to ten times (TNS in systemcases under "LC + DCTB" is worse than ours by 64 times) improvements on APDR, WNS, and TNS with much fewer violations, respectively. For the PLG scheme, in addition, our placer outperforms the other approaches (ranging from 1.7 to 5 times improvements on different metrics); the results show that our placement algorithm consistently improves the solution quality, on top of the proposed latch-clustering algorithm.

We also see much smaller clock wirelengths with 3–5% overheads in the signal wirelength, implying that our approach generates a placement result with good tradeoffs between signal and clock wirelengths.

C. Assessment of Latch Grouping

In this experiment, we analyze the potential impact derived from the latch grouping algorithms. We analyzed the final

TABLE III
COMPARISONS AMONG (ONLY) PLACEMENT ALGORITHMS UNDER THE LATCH-CLUSTERING SCHEME LC

Circuit	LC + Encounter									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
Norm.	–	–	3.03	1.72	2.35	4.41	–	0.96	2.75	1.01
Circuit	LC + DCTB									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
Norm.	–	–	2.75	1.96	3.08	10.89	–	0.94	2.38	0.99
Circuit	LC + Our Placer									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	14	0	3.43	22.16	15.55	67.53	0	13.15	1.52	14.67
aes_core	22	0	8.65	25.89	27.07	154.78	0	34.09	0.59	34.68
des_perf	25	0	6.02	46.37	14.56	135.23	0	139.10	7.59	146.70
mem_ctrl	127	0	9.55	51.73	49.93	352.67	0	18.95	1.35	20.31
pci_bridge32	35	0	8.54	44.11	48.74	421.84	0	32.61	3.95	36.55
s38417	40	0	3.16	18.18	2.30	31.66	0	10.51	0.69	11.19
systemcases	2	0	1.73	13.76	13.03	13.03	0	21.29	0.24	21.53
wb_conmax	398	0	12.17	68.33	123.16	4340.54	0	124.46	1.18	125.63
Norm.	–	–	1.00	1.00	1.00	1.00	–	1.00	1.00	1.00

The values in row "Norm." are all normalized to those of "LC + Our Placer."

TABLE IV
COMPARISONS AMONG (ONLY) PLACEMENT ALGORITHMS UNDER THE LATCH-CLUSTERING SCHEME PLG PROPOSED IN THIS PAPER

Circuit	PLG + Encounter									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	20	1	1.07	9.50	2.82	13.77	0.45	12.76	1.64	14.40
aes_core	11	1	1.21	5.75	7.31	25.06	0.03	32.76	0.37	33.12
des_perf	17	3	0.97	8.86	8.86	57.49	0.13	128.47	6.44	134.90
mem_ctrl	11	0	1.09	7.05	6.64	12.47	0	18.71	0.82	19.52
pci_bridge32	88	0	0.94	4.79	10.41	440.23	0	27.32	2.24	29.56
s38417	25	65	1.54	42.08	9.99	74.29	57.27	10.36	1.14	11.50
systemcases	2	14	3.90	12.12	10.14	10.14	2.34	22.08	1.01	23.09
wb_conmax	55	5	1.49	15.21	44.23	368.10	1.98	109.91	0.77	110.68
Norm.	–	–	1.71	2.55	(2.42)	(2.01)	–	0.97	2.16	1.00
Circuit	PLG + DCTB									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	23	3	1.26	7.17	16.34	70.60	0.36	12.62	1.68	14.30
aes_core	14	18	4.45	29.49	19.52	106.95	6.83	31.94	0.70	32.64
des_perf	31	136	3.03	100.00	30.85	168.54	164.42	122.74	9.83	132.57
mem_ctrl	29	48	5.16	33.19	12.54	31.03	33.17	19.39	1.61	21.00
pci_bridge32	234	54	2.53	18.94	35.16	1349.32	15.02	27.70	3.32	31.02
s38417	15	22	2.19	11.94	3.75	24.48	5.55	9.59	1.30	10.89
systemcases	1	17	4.08	14.43	6.77	6.77	4.64	18.62	0.99	19.61
wb_conmax	107	28	4.16	59.29	103.52	1215.28	20.76	124.34	1.14	125.47
Norm.	–	–	3.75	5.95	(4.46)	(5.42)	–	0.95	2.92	1.00

Note that normalized WNS/TNS values in "PLG + Encounter" and "PLG + DCTB" are in parentheses due to the denominator "0" in our placer.

wirelength results by the LC algorithm under different PGL-group topologies. In [16], the same skew values are assigned to each latch, implying that latches are arbitrarily grouped; the modified netlist that includes generators is then applied to the placer. By varying the number of latches that are grouped, the results could be quite different. Fig. 13 summarizes the normalized results. The number of generators and wirelength increase are all normalized to those in PLG. As shown in the figure, as the number of latches in a group increases, the number of required generators decreases while the wirelength overhead increases significantly. This phenomenon implies that fewer groups could restrict the freedom of a placer, which incurs substantial wirelength degradation. This experiment also shows that a simple (grouping) algorithm, like LC, is not suitable for the placement of pulsed-latches.

D. Comparisons Between Net-Weighting and Our Approach

Since our objective is to reduce the load capacitance by reducing wirelength between a pulse generator and pulsed latches, one simple alternative approach is net weighting. Like traditional clock-gating-aware placement or power-aware placement, we can place some blocks closer by imposing

TABLE V
PULSE-DEGRADATION COMPARISONS BETWEEN THE NET WEIGHTING AND OUR PLACEMENT ALGORITHM

Circuit	Net Weighting									
	TPV	LGV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	EWLR	Wirelength ($\times 10^7$)		
								SW	CW	TW
ac97_ctrl	16	2	1.36	7.38	8.99	31.98	0.62	12.40	1.29	13.69
aes_core	7	0	1.08	3.90	3.47	10.90	0	34.21	0.25	34.47
des_perf	12	28	0.97	36.05	39.55	76.32	6.34	127.45	4.97	132.42
mem_ctrl	6	0	0.87	5.53	5.38	12.67	0	20.54	0.66	21.20
pci_bridge32	77	0	1.20	4.61	13.32	448.67	0	27.63	1.79	29.41
s38417	27	0	0.77	2.09	2.96	51.63	0	10.52	0.58	11.10
systemcases	2	0	0.78	3.71	11.43	11.43	0	20.39	0.38	20.77
wb_conmax	40	21	2.73	21.98	42.39	443.09	4.87	120.11	0.91	121.02
Norm.	–	–	1.31	1.78	(2.72)	(2.21)	–	0.99	1.51	1.00

Note that the normalized WNS/TNS values in net weighting are in parentheses due to the denominator "0" in our approach.

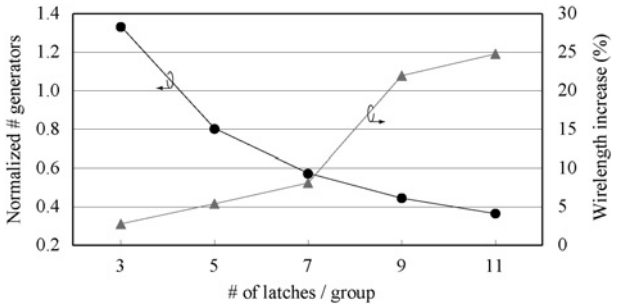

Fig. 13. Normalized number of generators and the total wirelength increase of LC (compared to PLG).

larger net weights among them. To explore the impact of net-weighting technique on pulsed-latch placement, we conducted an experiment to compare the results. In this experiment, we use the PLG algorithm to determine PGL groups and then apply net weighting during global placement, so that we see the real impact between different (pure) placement schemes. Table V summarizes the results between net weighting and our approach. Note that the detailed values of our approach can be found in Table II.

For the net weight on a PGL group w_g, we apply a form similar to [9] as follows:

$$w_g = \begin{cases} 1 + W \times \frac{C_g - T}{MC - T}, & \text{if } MC > C_g > T \\ 1, & \text{if } C_g \leq T \\ 1 + W, & C_g \geq MC \end{cases} \quad (17)$$

where W controls the scale of increased net weight, and C_g, MC, T denote the load capacitance of the PGL group, the maximum load capacitance among all groups, and the capacitance threshold for increased weight, respectively. In our implementation, we set W as 3 and $T = 0.3 \times MC$.

Table V indicates that our proposed approach is superior to net weighting in all parameters on average (except for SW). The reason can be understood by the unstability problem derived from the block density in net-weighting approach. Due to a larger weight, latches are placed closer to a generator; however, these latches might cause another density problem around the pulse generator, which potentially increases the burden for the placer to spread cells (including latches and other logics) more to satisfy the density constraint. In addition, the net-weighting technique does not have any guarantee to meet the load capacitance constraints during optimization. Consequently, using the net-weighting approach, we may find several groups violating the (MAWC) constraints during global placement iterations, which distorts pulse signals. However, in our proposed approach, since we provide a *smooth* mathematical optimization function for an analytical placer, the placer can optimize the three terms consistently (see Figs. 9 and 10).

E. Impact of Latch-Group Refinement

To determine PGL groups, the PLG algorithm is proposed in Section III-A with a timing-driven initial placement. We applied Encounter to generate such a timing-driven initial placement. We then determined PGL groups by the proximity of latches in the same pulse width. The determined PGL groups are used to guide the global placement for timing-integrity optimization.

However, the solution space could be limited if PGL-group settings are fixed after performing the PLG algorithm. Therefore, in this experiment, we explore additional approaches for PGL-group refinement. After performing the PLG algorithm, we modify the connections between latches and a generator based on the placement with more detailed information (the global placement result in our implementation), which might move some latches to other PGL groups. The main objective is to further improve the timing integrity. For simplicity, we refer to the latch-group refinement after PLG as the GR algorithm in the following discussion.

We modify PGL topologies based on global placement when the density overflow is less than 30%. The refinement of PGL groups is accomplished through moving latches from a "source" group to a "destination" group. The source group is defined as the group with an excessive load, while the destination group is the one with a smaller load. For PGL groups with the same type of pulse widths, we first sort them according to current wirelegnth $W_g(\mathbf{x}, \mathbf{y})$ as defined in previous sections in non-increasing order. In our implementation, we select the PGL groups with the top 25% largest $W_g(\mathbf{x}, \mathbf{y})$ as the source groups, while the groups with the last 25% smallest $W_g(\mathbf{x}, \mathbf{y})$ as the destination groups. For each source group, we sort the latches according to their distances to the generator in a queue Q_g. We then check each latch l_i from the beginning of Q_g one by one. To move a latch l_i, we choose a

TABLE VI
COMPARISONS BETWEEN THE LATCH-GROUP REFINEMENT (GR, AFTER PLG) AND PLG (ALONE) ALGORITHMS

Circuit	(PLG +) GR + Our Placer							
	TPV	APDR (%)	MPDR (%)	WNS (ps)	TNS (ps)	Wirelength ($\times 10^7$) SW	CW	TW
ac97_ctrl	26	1.55	5.87	7.25	87.83	14.74	1.26	16.00
aes_core	4	1.49	4.57	2.73	4.78	33.11	0.27	33.37
des_perf	6	1.34	6.57	2.61	9.32	140.00	5.98	145.98
mem_ctrl	3	1.69	5.98	2.69	3.24	20.88	0.65	21.53
pci_bridge32	126	1.39	5.70	18.30	658.65	31.57	1.93	33.49
s38417	18	1.72	6.32	3.68	26.33	11.07	0.85	11.92
systemcases	1	1.29	6.48	1.07	1.07	19.49	0.39	19.88
wb_conmax	29	1.74	5.32	37.63	216.84	128.12	0.34	128.46
Norm.	–	1.70	1.15	(1.67)	(2.26)	1.05	1.39	1.06

Note that we do not report the comparisons of LGV due to all zero values for both algorithms derived from barrier method.

destination group with the shortest distance between l_i and its corresponding generator. To prevent excessive load increase in the destination group, the total load of the destination group for l_i should be less than $\eta \times C_{\text{avg}}$, where C_{avg} is the average load of PGL groups of the same type, and η controls the tolerable increased load (we set $\eta = 75\%$ in the experiment). If the destination group does not satisfy η, we check the next destination group with the second shortest distance from l_i. The similar process continues to find a feasible destination group. If no destination groups can satisfy η, we terminate the refinement for the current type of pulse widths and repeat the above refinement again for another type of pulse widths.

After moving a latch from a source group to a destination group, we update the interconnection of the latch, the source generator, and the destination generator. We also update the MAWC of the source (destination) group by the wire load derived from the moved latch. After the group refinement on all pulsed-latches, we apply our global placement algorithm (see Fig. 11) on the updated netlist. Table VI summarizes the comparison between the GR and PLG algorithms. Note that the detailed values of "PLG alone" can be found in Table II.

From the experimental results, we find that the GR algorithm can further improve the WNS/TNS and also reduce timing violations on some cases. However, the GR algorithm also harms the solution for several cases. In particular, for systemcases, GR incurs additional timing failures compared to the PLG one. On average, GR has worse WNS (TNS) than PLG by 1.67 (2.26) times with a 39% clock wirelength overhead, which would increase dynamic power. As a result, the timing integrity resulted from GR cannot consistently be improved for all circuits; we discuss the potential problems as follows.

The reason could be two-fold. First, since the initial placement used for PLG is generated by Encounter with timing-driven mode. The initial placement considers wirelength implicitly, which gives a good initial reference to our PLG algorithm. Consequently, GR may not obtain significant improvements on general cases. Second, the refinement of GR mainly depends on the current wirelength of PGL groups; however, it is difficult to accurately estimate whether we can have better timing results after moving latches since there is a gap between wirelength and timing optimization. Note that such a gap exists in most wirelength-driven placement algorithms. One method to tackle this deficiency lies in the clock-network and placement co-synthesis. PGL groups can be refined iteratively along with the clock-network feedbacks; nevertheless, this topic is beyond the scope of this paper.

V. CONCLUSION

We introduced the placement problem of pulsed-latches for timing integrity. To tackle this problem, we proposed a better latch-group determination algorithm considering physical information and extended the analytical placement framework by a provably-good optimization force to reduce the potential load capacitance of pulse generators. Experimental results have shown the effectiveness and efficiency of our approach for pulsed-latch placement.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and anonymous reviewers for their very constructive comments.

REFERENCES

[1] C. Alpert, A. Kahng, G.-J. Nam, S. Reda, and P. Villarrubia, "A semi-persistent clustering technique for VLSI circuit placement," in *Proc. Int. Symp. Phys. Des.*, Apr. 2005, pp. 185–192.
[2] T. Baumann, D. Schmitt-Landsiedel, and C. Pacha, "Architectural assessment of design techniques to improve speed and robustness in embedded microprocessors," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2009, pp. 947–950.
[3] S. Boyd and L. Vandenberghe, *Convex Optimation*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[4] Cadence Design Systems, Inc. *SOC Encounter RTL-toGDSII System* [Online]. Available: http://www.cadence.com
[5] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie, "mPL6: Enhanced multilevel mixed-size placement," in *Proc. Int. Symp. Phys. Des.*, Apr. 2006, pp. 212–214.
[6] D. Chinnery and K. Keutzer, *Closing the Gap Between ASIC and Custom*. Dordrecht, The Netherlands: Kluwer Academic, 2002.
[7] C. Chu, "FLUTE: Fast lookup table based wirelength estimation technique," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 696–701.
[8] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 7, pp. 1228–1240, Jul. 2008.
[9] Y. Cheon, P.-H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "Power-aware placement," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2005, pp. 795–800.

[10] Y.-L. Chuang, S. Kim, Y. Shin, and Y.-W. Chang, "Pulsed-latch-aware placement for timing-integrity optimization," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2010, pp. 280–285.
[11] J. Cong, M. Romesis, and J. R. Shinnerl, "Fast floorplanning by look-ahead enabled recursive bipartitioning," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2005, pp. 1119–1122.
[12] Synopsys. *HSPICE* [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification / CircuitSimulation / HSPICE / Pages/default.aspx
[13] C. Albrecht. (2005, Jun.). *IWLS 2005 Benchmarks*, Cadence Berkeley Labs [Online]. Available: http://iwls.org/iwls2005/benchmarks.html
[14] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734–747, May 2005.
[15] S. Kim, I. Han, S. Paik, and Y. Shin, "Pulser gating: A clock gating of pulsed-latch circuits," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2011, pp. 190–195.
[16] H. Lee, S. Paik, and Y. Shin, "Pulse width allocation and clock skew scheduling: Optimizing sequential circuits based on pulsed latches," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 3, pp. 355–366, Mar. 2010.
[17] S. Lee, S. Paik, and Y. Shin, "Retiming and time borrowing: Optimizing high-performance pulsed-latch-based circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2009, pp. 375–380.
[18] H.-C. Li, M.-C. Chen, and K. Ho, "System and method for replacing flip-flops with pulsed latches in circuit designs," U.S. Patent 7 694 242, Apr. 6, 2010.
[19] *Nangate 45 nm Open Cell Library* [Online]. Available: http://www.nangate.com
[20] N. Viswanathan, M. Pan, and C. Chu, "FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2007, pp. 135–140.
[21] W. C. Naylor, R. Donelly, and L. Sha, "Nonlinear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," U.S. Patent 6 301 693, Oct. 9, 2001.
[22] OpenCores [Online]. Available: http://www.opencores.org
[23] M. Pan and C. Chu, "A step to integrate global routing into placement," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2006, pp. 464–471.
[24] 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, CA, Tech. Rep. UCB/ERL M92/41, May 1992.
[25] Y. Shimazaki and M. Nishibori, "Pulsed latch circuit and semiconductor integrated circuit," U.S. Patent 7 768 294, Aug. 3, 2010.
[26] S. Shibatani and A. H. C. Li, "Pulse-latch approach reduces dynamic power," *EETimes Online*, 2006.
[27] Y. Shin and S. Paik, "Pulsed-latch circuits: A new dimension in ASIC design," *IEEE Des. Test Comput.*, vol. 28, no. 6, Nov.–Dec. 2011.
[28] Y. Wang, Q. Zhou, X. Hong, and Y. Cai, "Clock-tree aware placement based on dynamic clock-tree building," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 2040–2043.
[29] C.-H. Wang, C.-C. Chen, C.-S. Tsai, and S.-Y. Ying, "Clock circuit and method for pulsed latch circuits," U.S. Patent 2010/0259308, Oct. 14, 2010.
[30] J.-S. Yim, S.-O. Bae, and C.-M. Kyung, "A floorplan-based planning methodology for power and clock distribution in ASICs," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 1999, pp. 766–771.
[31] K. Vorwerk and A. Kennings, "An improved multi-level framework for force-directed placement," in *Proc. Conf. Des. Autom. Test Eur.*, Mar. 2005, pp. 902–907.

Yi-Lin Chuang received the B.S. degree from National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2006, and the Ph.D. degree from National Taiwan University (NTU), Taipei, Taiwan, in 2011.

He is currently a Principle Engineer with the Design and Technology Platform, TSMC, Ltd., Hsinchu, Taiwan. His current research interests include combinatorial optimization with applications to very large scale integration (VLSI) design automation, VLSI placement systems, and power optimization.

Dr. Chuang received the Best Paper Nomination from ICCAD-2009 and the NTU Outstanding Research Award in 2010.

Sangmin Kim received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2008 and 2010, respectively. He is currently working toward the Ph.D. degree with the Department of Electrical Engineering, KAIST.

His current research interests include computer-aided design for low-power design and pulsed-latch application-specific integrated circuit design.

Youngsoo Shin received the B.S., M.S., and Ph.D. degrees in electronics engineering from Seoul National University, Seoul, Korea.

From 2000 to 2001, he was a Research Associate with the University of Tokyo, Tokyo, Japan, and from 2001 to 2004 he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY. He joined the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2004, where he is currently an Associate Professor. His current research interests include computer-aided design with emphasis on low-power design and design tools, high-level synthesis, sequential synthesis, and programmable logic.

Dr. Shin received several awards, including the Best Paper Award at the 2005 International Symposium on Quality Electronic Design and the 2002 IP Excellence Award from Japan. He has been a member of the technical program committees and organizing committees of many technical conferences, including DAC, ICCAD, ISLPED, ASP-DAC, CASES, ISVLSI, and ISCAS. He is a member of the ACM SIGDA Low Power Technical Committee and is serving as an Associate Editor of ACM TODAES.

Yao-Wen Chang (S'94–A'96–M'96) received the B.S. degree from National Taiwan University (NTU), Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas, Austin, in 1993 and 1996, respectively, all in computer science.

He is currently the Director of the Graduate Institute of Electronics Engineering and a Distinguished Professor with the Department of Electrical Engineering, NTU. His current research interests include very large scale integration physical design and design for manufacturability. He has been working closely with the industry in these areas. He has co-edited one textbook on electronic design automation and co-authored one book on routing and over 190 ACM/IEEE conference/journal papers in these areas.

Dr. Chang is a four-time winner of the ACM ISPD Contests (placement in 2006, global routing in 2008, clock network synthesis in 2009 and 2010) and a recipient of six Best Paper Awards (twice at ICCD, and so on) and 20 Best Paper Award Nominations from DAC (five times), ICCAD (four times), and so on in the past ten years. He has received many research/teaching awards, such as the Distinguished Research Award from the National Science Council of Taiwan (twice), the IBM Faculty Award, the CIEE Distinguished EE Professor, the MXIC Young Chair Professorship, and excellent teaching awards from NTU (seven times). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and an Editor of the *Journal of Information Science and Engineering*. He has served as the General and Steering Committee Chair of ISPD, and the Program Chair of ASP-DAC, FPT, and ISPD, and on the ICCAD Executive Committee, the ASP-DAC Steering Committee, the ACM/SIGDA Physical Design Technical Committee, the IEEE CEDA Conference Committee, and the technical program committees of ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SLIP, SOCC, and VLSI-DAT. He has served as an Independent Board Director of Genesys Logic, Inc., Taipei, a Technical Consultant of Faraday Technology Corporation, Hsinchu, Taiwan, MediaTek, Inc., Hsinchu, and RealTek Semiconductor Corporation, Hsinchu, the Chair of the EDA Consortium of the Ministry of Education, Taiwan, and a member of the Board of Governors of the Taiwan IC Design Society.