# Power-Driven Flip-Flop Merging and Relocation

Shao-Huan Wang, Yu-Yi Liang, Tien-Yu Kuo, and Wai-Kei Mak, *Member, IEEE*

***Abstract*—We propose a power-driven flip-flop (FF) merging and relocation approach that can be applied after conventional timing-driven placement and before clock network synthesis. It targets to reduce the clock network size and thus the clock power consumption while controlling the switching power of the nets connected to the FFs by selectively merging FFs into multibit FFs and relocating them under timing and placement density constraints. The experimental results are very encouraging. For a set of benchmarks, our approach reduced the switching capacitance of clock network by 36%–43% after gated clock tree synthesis. Finally, the total switching capacitance of clock network and nets connected to the FFs is reduced by 24%–29%.**

***Index Terms*—Clock network, flip-flop merging, low power, multibit flip-flop, postplacement optimization.**

## I. Introduction

CLOCK network plays an important role in power consumption as it accounts for up to 50% [1] of dynamic power in some circuits for its highest switching rate. Many kinds of clock power-reduction techniques have been proposed. References [2] and [3] worked on buffer sizing for clock power minimization. References [4]–[6] designed some new low-power flip-flop (FF) structures. References [7]–[9] discussed clock gating. References [10]–[12] minimized the power of clock network by considering the location of registers in the placement stage. They try to group registers into clusters and place registers in a cluster closer to reduce the wirelength (WL) of the leaf level of a clock tree. In this paper, we form multibit flip-flops (MBFFs) to reduce the overall clock power and switching power of signal nets connected to the FFs.

The use of MBFF was first proposed in [13] for reducing clock delay, controlling clock skew, and improving routing resource utilization. Reference [14] introduced a design methodology for MBFF inference during logic synthesis for area and power reduction. However, to form MBFFs in early design stage, it is hard to consider its effect on timing. So, recently a few works [15]–[18] considered reducing the power consumption of FFs by incrementally forming more MBFFs at the postplacement stage.

Fig. 1 shows an example of replacing two traditional FFs by a MBFF. Because of manufacturing ground rules in advanced process technology, inverters tend to be oversized so that an inverter can drive more than one traditional FF. By merging multiple 1-bit FFs into one MBFF, it is possible to eliminate some inverters. For example, we can eliminate two inverters after merging in Fig. 1. It will reduce the total area and power consumption of the FFs. This is the motivation for [15]–[18].

More importantly, using MBFF can also reduce the number of clock sinks. This will reduce the wirelength of the clock network and the buffers required in the clock network to maintain the slew and balance the skew. Therefore, power consumed by the clock network will be substantially reduced.

In this paper, we target to form MBFFs to simultaneously minimize the clock power and the signal net switching power instead of optimizing merely the FF power as in [15]–[18]. The clock power includes the switching power of the interconnect, buffers, clock gates, and FFs in the clock distribution network. We note that creating a smaller clock tree with smaller number of sinks by FF merging will reduce: 1) the switching power of clock interconnect (due to shorter clock tree wirelength); 2) the switching power of clock buffers and clock gates (due to reduction of number of clock buffers and clock gates required); and 3) the FF power (due to the replacement of 1-bit FFs by more power-efficient MBFFs). Hence, we focus on minimizing the number of clock sinks by FF merging.

We propose a power-driven FF merging and relocation approach that can be applied after conventional timing-driven placement and before clock network synthesis. To apply FF merging after initial placement, the following issues need to be considered: decide which FFs should be merged into MBFFs and decide their locations considering placement density, timing constraints, and switching power of signal nets to obtain the best possible power saving. There are a lot of FFs on a chip at different locations but we cannot merge every FF due to timing constraint. Moreover, there are many possible combinations of FFs and each will result in different power consumption. Since a library offers many kinds of MBFF, in terms of number of bits, area, and capacitance, we have to find the target MBFFs we should use. Besides, without careful planning, MBFFs that are generated may be put into new places that may cause area overflow and routing congestion.

The contributions of our paper is as follows. First, we show that power-driven FF merging is a NP-hard problem. Second,

Manuscript received June 13, 2011; revised August 12, 2011 and October 5, 2011; accepted October 26, 2011. Date of current version January 20, 2012. This work was supported in part by the National Science Council, under Grant NSC 100-2221-E-007-088. This paper was recommended by Associate Editor J. Hu.

The authors are with the Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: shwang@cs.nthu.edu.tw; yuyiliang@cs.nthu.edu.tw; u9562105@oz.nthu.edu.tw; wkmak@cs.nthu.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.2011.2177460

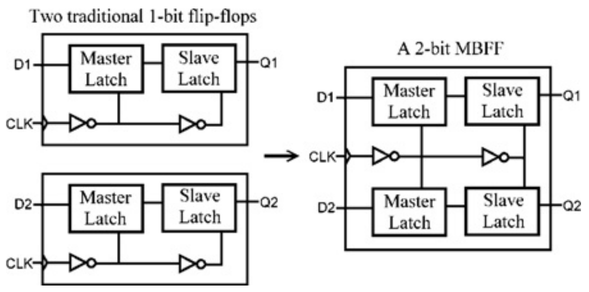

Fig. 1. Replacing two traditional FFs by a 2-bit MBFF.

we present an effective heuristic approach to simultaneously minimize the clock power and the signal net switching power. Our approach is based on the efficient computation of all maximal cliques in a rectangle intersection graph and a simple but effective sampling technique. Extensive experiments show that our approach is robust and it produces high-quality results in reasonable amount of time.

Compared with our preliminary version [19], we improved and parallelized the procedure of finding all maximal cliques and provide the algorithmic details in Section IV-A. In addition, a faster and more sophisticated method to generate a $k$-bit MBFF from a clique is described in Section IV-B. The experimental results are completely updated since we can take more samples in the same amount of time after reducing the runtime of finding all maximal cliques and extracting MBFFs. Besides, more and different experiments are done to show the robustness and effectiveness of the proposed approach including the usage of preoptimized benchmarks as the new baseline and experiments with gated clock tree synthesis (CTS) to show the reduction in the clock tree's switched capacitance, clock wirelength, enable net wirelength, number of buffers, and clock gates. Finally, we included a new discussion section that discusses the adaptation of our method to multiple power domains and analyzes the effects of varying our parameters.

The remainder of this paper is organized as follows. We define the power-driven FF merging and relocation problem in details in Section II. We define some useful terms and give the proof of NP-hardness of the problem in Section III. Section IV contains the detailed descriptions of the proposed algorithm stage by stage. The experimental results are presented in Section V. Discussions and conclusions are in Sections VI and VII, respectively.

## II. Problem Formulation

We want to reduce the switching power of the clock network and the signal nets by selectively relocating and merging FFs into MBFFs at the postplacement stage. The switching power of a signal $i$ is given by $k\alpha_i C_i V^2$, where $k$ is a constant, $\alpha_i$ is the switching rate, $C_i$ is the capacitance to be charged and discharged, and $V$ is the supply voltage.

Before clock network synthesis, the wirelength and, hence, the capacitance of the clock network are not known. However, our experiment shows that the reduction percentage of the number of sinks is roughly proportional to the reduction percentage of the wirelength of clock network. In addition, [10] observed that most of the wire capacitance of a clock tree is at the leaf level. In some industrial designs, the percentage can reach 40%. Hence, reducing the number of clock sinks can result in a fine level of clock power saving due to the reduction of clock interconnect capacitance and the reduction of buffers and clock gates required. However, merging a group of FFs into a single MBFF will affect the wirelength and hence the net switching power of the signals connected to these FFs. The clock power reduction by FF merging may be canceled out by the change in switching power of signal nets. So, we have to consider them simultaneously. We define the postplacement power-driven FF merging and relocation problem as follows.

*Input:* We are given a preplaced design with a set of FFs[1] and their corresponding fanin and fanout locations, and a MBFF library. We are also given some placement density constraint and timing constraint.

*Objective:* Selectively relocate and merge FFs into MBFFs to minimize the number of clock sinks and signal net switching power subject to the given placement density and timing constraints.

We must ensure that the design meets certain constraints after FF merging. The first one is timing constraint. When multiple 1-bit FFs are merged into a MBFF, the new MBFF may incur additional routing length due to the change of the location, and thus changes the timing. Fig. 2 shows an example. But we can take advantage of the original timing slacks between a FF and its fanin and fanout pins.

The timing slack between a FF and a pin can be converted into a maximum distance constraint from the pin. For simplicity, we can distribute the timing slacks as shown in Fig. 3. If $D_{AB}$ is the delay of the combinational logic between two pins $A$ and $B$ connected to FFs $\text{FF}_A$ and $\text{FF}_B$. We allocate half of the total slack as the timing slack between pin $A$ and $\text{FF}_A$, and the other half as the timing slack between pin $B$ and $\text{FF}_B$. Then, we use (1) [18] to translate the timing slack to a corresponding wirelength under the Elmore delay model as follows:

$$l \le \frac{\sqrt{c_0^2R^2 + r_0^2C^2 + 2r_0c_0t_{\max}} - c_0R - r_0C}{r_0c_0} \qquad (1)$$

where $r_0$ is the unit resistance, $c_0$ is the unit capacitance, $R$ is the driver strength, $C$ is the driving load, and $t_{\max}$ is the timing slack between a pin and its corresponding FF. We note that the Elmore delay model can be replaced by any other timing model if desired. Thus, the feasible region that a FF can be placed in would be a 45° tilted rectangular region, as shown in Fig. 4.

We also consider the placement density of each local area to avoid routing congestion. Therefore, we partition a layout into $b_N \times b_M$ bins, and each bin has a maximum placement density constraint. We assume every MBFF can only be placed on an unoccupied grid point within a bin without violating the placement density constraint.

[1]They are not necessary to be 1-bit. Our algorithm can also handle original FFs with multiple bits.

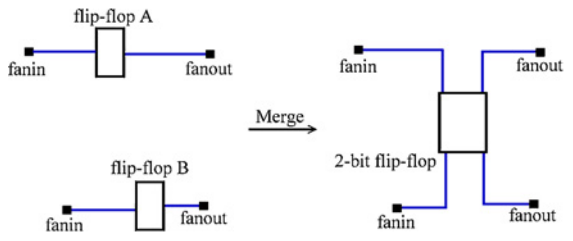

Fig. 2. Wirelength and timing will be changed by merging.

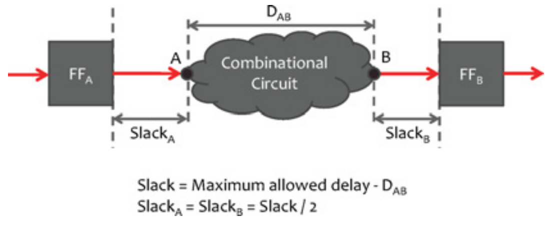
D_AB
FF_A
A
Combinational Circuit
B
FF_B
Slack_A
Slack_B
Slack = Maximum allowed delay - D_AB
Slack_A = Slack_B = Slack / 2

Fig. 3. Timing slack of pins connected to FFs.

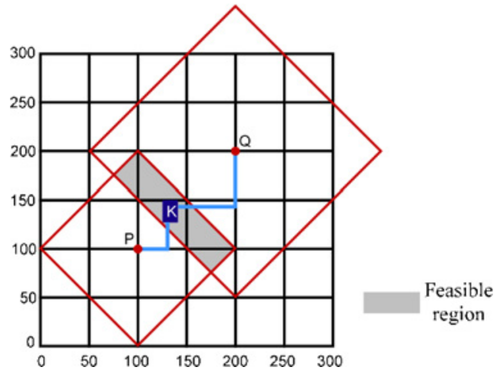

Fig. 4. Feasible region of FF K, which is connected to pins P and Q. The maximum allowable distances from P and Q to K are assumed to be 100 and 150, respectively.

## III. Preliminaries

The feasible region of each original FF is a 45° tilted rectangular region, and these feasible regions may intersect each other. Every intersected region represents the feasible region of a MBFF for the corresponding original FFs. As shown in Fig. 5, each intersection of feasible regions of original FFs is a clique in the corresponding rectangle intersection graphs.[2] Therefore, if we consider the case to minimize the number of clock sinks with timing constraint only, the problem is exactly the same as minimum clique partition of a rectangle intersection graph (MCPRIG). The MCPRIG problem is defined as follows.

*Input:* Given a rectangle intersection graph $G = (V, E)$. $V$ is the vertex set representing all rectangles and $E$ is the edge set representing all intersection relations.

*Objective:* Find $k$ disjoint cliques to cover all the vertices in $V$, where $k$ is minimum.

MCPRIG has been proved to be NP-hard in [20]. And a MCPRIG instance can be reduced to an instance of the power-

[2]In a rectangle intersection graph, each vertex corresponds to a rectangle, and if two rectangles intersect with each other, there is an edge between the corresponding vertices.

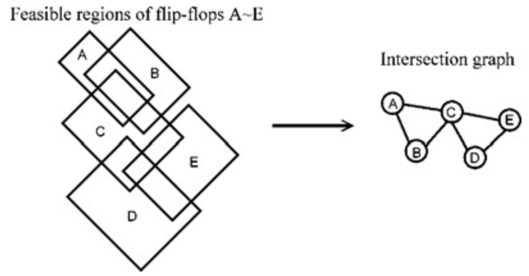

Fig. 5. Intersection graph.

driven FF merging and relocation problem of minimizing the number of clock sinks with timing constraint only by noting that each rectangle in a rectangle intersection graph can always be regarded as the feasible region of a FF formed by the common region of a pair of squares on a plane. Hence, we get the following theorem.

*Theorem 1:* The power-driven FF merging and relocation problem is NP-hard.

Although MCPRIG is NP-hard, it still has some property which is helpful to note for us to design an algorithm for the power-driven FF merging and relocation problem. We define some terms and introduce the property as follows.

*Definition 1:* A maximal clique is a clique that cannot be extended by including one more adjacent vertex.

*Definition 2:* For a vertex $v_i \in V$, participating degree of $v_i$ is the number of maximal cliques which cover $v_i$.

*Definition 3:* A vertex $v_i$ is a critical vertex if and only if the participating degree of $v_i$ is 1.

*Definition 4:* A maximal clique $c_i$ is called an essential maximal clique if and only if $c_i$ contains at least one critical vertex.

*Lemma 1:* Let $C_e$ be the set of all essential maximal cliques. $|C_e|$ is a lower bound of $k$. In other words, $k \geq |C_e|$.

*Proof:* By the definition of essential maximal clique, we know there exist $|C_e|$ critical vertices covered by different maximal cliques. Hence, in order to cover these vertices we need at least $|C_e|$ cliques. ■

Lemma 1 gives a lower bound of the number of sinks of our merging problem. Therefore, if we want to minimize the number of sinks, it is good to start from finding all maximal cliques. In the next section, we will use the concept of participating degree to design an efficient algorithm for the power-driven FF merging and relocation problem.

## IV. Power-Driven FF Merging and Relocation Algorithm

The proposed design flow is shown in Fig. 6. First, we compute the feasible region of each original FF according to the timing slacks after placement. Second, we form the corresponding rectangle intersection graph and compute all the maximal cliques in order to get all groups of FFs that can be merged without violating any timing constraint (Section IV-A). The third step is to find a set of nonconflicting[3] MBFFs

[3]Nonconflicting means if one FF has been merged into some MBFF, then this FF cannot be merged into other MBFF.

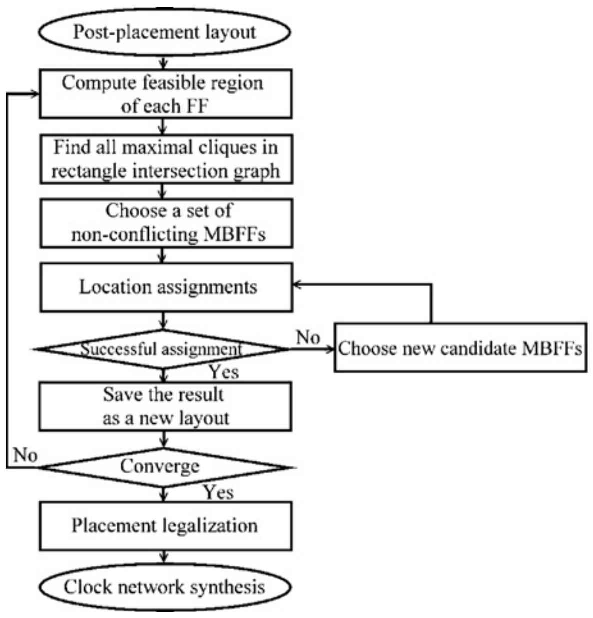
Post-placement layout
Compute feasible region of each FF
Find all maximal cliques in rectangle intersection graph
Choose a set of non-conflicting MBFFs
Location assignments
Successful assignment
No
Choose new candidate MBFFs
Yes
Save the result as a new layout
No
Converge
Yes
Placement legalization
Clock network synthesis

Fig. 6. Design flow with power-driven FF merging and relocation.

by dividing maximal cliques into MBFFs (Section IV-B). The last step is to determine the location of each MBFF (Section IV-C). Noticeably, there might exist 1-bit MBFFs in the set of nonconflicting MBFFs. They can also be relocated to further optimize the switching power of signal nets.

Because our algorithm can take care of original FFs which are more than 1-bit, we can apply our algorithm iteratively until there is no further improvement. We note that, although we assign MBFFs to unoccupied grid points subject to placement density constraints, there may still exist some overlappings between the MBFFs and preplaced cells. Placement legalization can be done using techniques like [21] and [22].

## A. Mergeable FF Groups Determination

Each group of FFs that can be merged without violating any timing constraint corresponds to a clique in the rectangle intersection graph. There can be a large number of cliques in the rectangle intersection graph. But we do not have to explicitly generate and store all of them as they are all contained in the maximal cliques. So we will just find all maximal cliques.

Finding a maximum clique in a general graph is NP-hard, but to find a maximum clique or all maximal cliques in a rectangle intersection graph is in P. Reference [23] presented an algorithm to find a maximum clique in $O(nlog(n))$ time. Here, we utilize the sweep line method [24] to find all maximal cliques. Sweep line method is a kind of scanning to determine the relation between rectangles on a plane. It keeps track of in-edges and out-edges to determine whether we are now inside a particular rectangle or not.

To perform the method, we rotate the entire chip by 45°. Then, we extend all vertical boundaries of feasible regions. Every two adjacent extended boundaries form a strip. We call the top and bottom boundaries of a feasible region an in-edge and an out-edge, respectively; see Fig. 7.

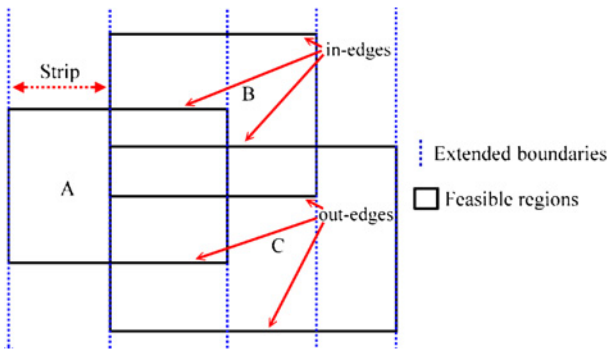

Fig. 7. Feasible regions of FFs A, B, and C after rotation.

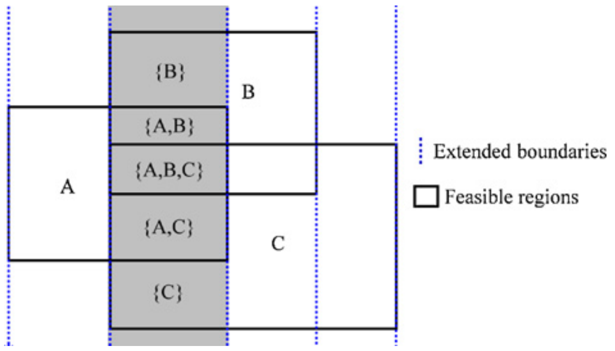

Fig. 8. We can find the maximal clique {A, B, C} in the second strip from the left.

For a maximal clique in each strip, its size is greater than the sizes of the cliques above and below it in the same strip. As shown in Fig. 8, in the second strip from the left, the size of clique {A, B, C} is greater than cliques {A, B} and {A, C}. Thus, we use this property to find all maximal cliques in a strip $\mathcal{S}$ as shown in Procedure 1. In lines 5–7 of Procedure 1, if we encounter an in-edge, this means that we enter a feasible region and we put it into set $c$. In lines 9–12, if we encounter an out-edge after successive in-edges, the feasible regions in set $c$ form a maximal clique. Besides, encountering an out-edge means we leave a feasible region and hence we remove such feasible region from $c$ as line 13.

Assume $n$ is the number of rectangles, then there are at most $O(n^2)$ maximal cliques in the rectangle intersection graph. Here, we assume $m = O(n)$ is the maximum size of any clique. Thus, the time complexity for storing all maximal cliques in a rectangle intersection graph is $m \times O(n^2) = O(n^3)$. In our algorithm, we take all the $x$-coordinates of vertical edges and perform sorting to get the contiguous strips. This step takes $O(nlog(n))$ time for sorting and we will get at most $O(2n) = O(n)$ strips. Then, we use a binary search tree to perform the operation of insertion and deletion of all horizontal edges for generating the set $\mathcal{E}$ in Procedure 1. By applying the sweep line algorithm, each edge will be inserted one time and deleted one time. Therefore, insertion and deletion of all horizontal edges takes $O(nlog(n))$ time. In sweep line process, we handle each strip as in Procedure 1. For efficiency, the clique $c$ in Procedure 1 is implemented by a binary search tree. Thus, the complexity of Procedure 1 is $O(nlog(n))$ and the total complexity for $O(n)$ strips is $O(n^2log(n))$.

**Procedure 1** Find_Max_Cliques ($\mathcal{S}$)

```
Input: A strip $\mathcal{S}$.
Output: The set of all maximal cliques $\mathcal{C}$ in strip $\mathcal{S}$.
1: // $\mathcal{E}$ is a set of edges that contains all in-edges and out-edges in strip $\mathcal{S}$.
2: // $f(e)$ is the feasible region that contains edge $e$.
3: $c \leftarrow \phi$; $\mathcal{C} \leftarrow \phi$; $flag \leftarrow False$
4: for each edge $e$ in $\mathcal{E}$ from top to bottom do
5:    if $e$ is in-edge then
6:       $c \leftarrow c \cup \{f(e)\}$
7:       $flag \leftarrow True$
8:    else {$e$ is out-edge}
9:       if $flag$ is $True$ then
10:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
11:         $flag \leftarrow False$
12:      end if
13:      $c \leftarrow c \backslash \{f(e)\}$
14:   end if
15: end for
16: return $\mathcal{C}$
```

Note that the maximal cliques in a strip might not be the maximal cliques in the rectangle intersection graph. As shown in Fig. 8, $\{A\}$ is a maximal clique in the first strip from the left. However, $\{A\}$ is a subset of the maximal clique $\{A, B, C\}$ in the second strip from the left. It means that $\{A\}$ is not a maximal clique in the rectangle intersection graph. If we have a set of maximal cliques from strip $S_1$ to $S_{i-1}$ and a set of maximal cliques is found in strip $S_i$, we will merge and remove any maximal clique in $S_1$ to $S_{i-1}$ which is a subset of some clique in $S_i$, and vice versa. Thus, we can get the maximal cliques from $S_1$ to $S_i$. By induction, we can find all the maximal cliques in the rectangle intersection graph. When Procedure 1 finds a maximal clique in $S_i$, it is easy to prove that one of its subcliques can be in $S_1$ to $S_{i-1}$ but not any other. For efficiency, we use a hash table to store all the maximal cliques from $S_1$ to $S_{i-1}$. Thus, for each maximal clique found in $S_i$, it takes $m = O(n)$ time to get the hash key and remove the desired clique in $S_1$ to $S_{i-1}$. By the above process, we can find all maximal cliques in the rectangle intersection graph in $O(n^3)$ time which is the same as the complexity of storing all maximal cliques.

To parallelize this algorithm, we divide the chip into $p$ regions of at most $\lceil \frac{2n}{p} \rceil$ strips each for $p$ cores and find the maximal cliques in each region independently.

### *B. MBFF Extraction Considering Participating Degree*

In this stage, we want to extract MBFFs from the maximal cliques according to the given library. The main idea of MBFF extraction is to create MBFFs with the smallest cost greedily. First, for each maximal clique, we generate a set of possible MBFFs. Then, for each set, we take the one with smallest cost as the candidate MBFF of the corresponding maximal clique. Second, we put the one with the smallest cost from these candidate MBFFs into the set of nonconflicting MBFFs, said $\mathcal{N}$, repeatedly. Each time we put a MBFF $\beta$ into $\mathcal{N}$, we must make sure that $\beta$ does not conflict with any existing MBFF belonging to $\mathcal{N}$. After that, we create a new candidate MBFF from the clique $c$ which generated $\beta$. This new candidate MBFF also should have no conflict with any existing MBFF belonging to $\mathcal{N}$. After the set of candidate MBFFs becomes empty, $\mathcal{N}$ is the target set of nonconflicting MBFFs to do location assignment.

Since, we want to avoid creating a MBFF whose fanins and fanouts are too far away to reduce the total wirelength of signal nets. The cost of a MBFF $\beta$ is defined as follows:

$$\text{cost}(\beta) = \frac{\mathcal{D}(\beta)}{\mathcal{B}(\beta)} + \lambda \times \frac{\sum_{\text{net}_i \in \text{Net}(\beta)} (\alpha_i \times \text{EstWL}_i)}{\sum_{\text{net}_i \in \text{Net}(\beta)} (\alpha_i \times \text{OWL}_i)} \quad (2)$$

where $\mathcal{D}(\beta)$ is the average participating degree[4] of MBFF $\beta$, and $\mathcal{B}(\beta)$ is the total number of bits of $\beta$. Net($\beta$) is the set of all the signal nets connected to $\beta$, and $\text{EstWL}_i$ is the estimated wirelength of $\text{net}_i$. We take the weighted median values of the $x$-coordinates and of the $y$-coordinates of all pins connected to $\beta$ to estimate the final location of $\beta$ to get $\text{EstWL}_i$ for a $\text{net}_i$ connected to $\beta$.

The intuition for defining cost($\beta$) as in (2) is as follows. We use $\frac{\mathcal{D}(\beta)}{\mathcal{B}(\beta)}$ as a measure of the effect on the final total number of sinks if MBFF $\beta$ is formed. First, if the number of bits of $\beta$ is larger, it means we can reduce more sinks when MBFF $\beta$ is formed. So, we have $\mathcal{B}(\beta)$ in the denominator of $\mathcal{D}(\beta)/\mathcal{B}(\beta)$. Second, the smaller the participating degree of an original FF, the less merging choices it has. So, a MBFF $\beta$ with smaller average participating degree should be given higher priority to be chosen which leads to the numerator in $\mathcal{D}(\beta)/\mathcal{B}(\beta)$.

However, finding all the possible MBFFs contained in a clique takes exponential time and memory space. For example, if a clique $c$ is formed by eight 1-bit FFs, and the library only supports 2-bit MBFF, then $\binom{8}{2}$ different 2-bit MBFFs can be generated from $c$. Instead, we generate $\min(\binom{|c|}{k}, S_{\text{bound}})$ $k$-bit MBFFs randomly for clique $c$, where $S_{\text{bound}}$ is a constant value of sampling bound. In our experiments, we set $S_{\text{bound}}$ to 50. Then, we calculate the cost of each one and take the one with the smallest cost as a candidate MBFF of the clique. We call this operation Best_Sample(c) as shown in Procedure 2.

A naive method to generate a $k$-bit MBFF from a clique of size $q$ is by randomly selecting a FF in the clique many times until we get $k$ distinct FFs to form a $k$-bit MBFF. Although the probability distribution of this sampling method is uniform, the time to generate a MBFF sample is nondeterministic and the cycle length of random generator may not be long enough to keep the uniform distribution of MBFF samples. Thus, we propose to randomly generate a single integer $i$ in the range $[1, \binom{q}{k}]$ using Mersenne Twister [25] in line 5 of Procedure 2, and then using an encoding-based procedure $I$th_Combination($c, q, k, i$) described in Procedure 3 to pick the $i$th encoded $k$-bit MBFF from a clique of size $q$. For example, there are four FFs in clique $c = \{A, B, C, D\}$ and we would like to generate a 2-bit MBFF sample from $c$. We encode all the MBFF samples in lexicographical order as in Table I. $I$th_Combination($c$, 4, 2, 3) will return the third encoded 2-bit MBFF from $c$, i.e., $\{A, D\}$. It is easy to see that Procedure 3

[4]The average participating degree is defined as the average of participating degree of all constituent FFs in $\beta$.

**Procedure 2** Best_Sample($c$)

```
Input: A clique $c$.
Output: The best MBFF in all samples.
1: $q \leftarrow size(c)$
2: $best \leftarrow \phi$
3: for all $k$ s.t. $k \leq q$ and there is a $k$-bit MBFF in library do
4:   for $times := 1$ to $\min(\binom{q}{k}, S_{\text{bound}})$ do
5:     $i \leftarrow$ a random value in $[1, \binom{q}{k}]$
6:     $sample \leftarrow$ $I$th_Combination$(c, q, k, i)$
7:     if $cost(sample) < cost(best)$ then
8:       $best \leftarrow sample$
9:     end if
10:  end for
11: end for
12: return $best$
```

**Procedure 3** $I$th_Combination$(c, q, k, i)$

```
Input: A clique $c$ with size $q$.
       Positive integers $k$ and $i$ s.t. $k \leq q$ and $i \leq \binom{q}{k}$.
Output: The $i$th encoded $k$-bit MBFF $\beta$ of $c$.
1: $q' \leftarrow q; k' \leftarrow k; i' \leftarrow i$
2: $\beta \leftarrow \phi$
3: for $j := 1$ to $q$ do
4:   if $k' \leq 0$ then
5:     break
6:   end if
7:   if $i' \leq \binom{q'-1}{k'-1}$ then
8:     $\beta \leftarrow \beta \cup \{c[j]\}$
       //$c[j]$ is the $j$th FF in clique $c$.
9:     $k' \leftarrow k' - 1$
10:  else
11:    $i' \leftarrow i' - \binom{q'-1}{k'-1}$
12:  end if
13:  $q' \leftarrow q' - 1$
14: end for
15: return $\beta$
```

TABLE I
ENCODING OF 2-BIT MBFFS FROM $\{A, B, C, D\}$

| Encoding | MBFF |
|---|---|
| 1 | $\{A, B\}$ |
| 2 | $\{A, C\}$ |
| 3 | $\{A, D\}$ |
| 4 | $\{B, C\}$ |
| 5 | $\{B, D\}$ |
| 6 | $\{C, D\}$ |

takes $O(q)$ time and Procedure 2 takes $O(L \cdot S_{\text{bound}} \cdot q)$, where $L$ is the MBFF library size. Since $L$ and $S_{\text{bound}}$ are constants, the time complexity of Procedure 2 is $O(q)$.

The MBFF extraction algorithm is shown in Algorithm 1. In lines 5–7, we generate one candidate from each maximal clique and push it into a heap. From lines 9–14, we pick the best candidate $\beta$ from the heap and put it into $\mathcal{N}$ if it does not conflict with any MBFF belonging to $\mathcal{N}$. And in lines 15–17, we generate another candidate by sampling the maximal clique

**Algorithm 1** MBFF extraction

```
Input: The set of all maximal cliques $\mathcal{C}$.
Output: A set of nonconflicting MBFFs $\mathcal{N}$.
1: // $g(\beta)$ is the clique which generated MBFF $\beta$.
2: // $\mathcal{P}$ is the set of FFs allocated into MBFFs in $\mathcal{N}$.
3: // $H$ is a heap of candidate MBFFs. The key value is the
   cost of each MBFF.
4: $\mathcal{N} \leftarrow \phi$; $\mathcal{P} \leftarrow \phi$; $H \leftarrow \phi$
5: for all $c \in \mathcal{C}$ do
6:   $H$.Push(Best_Sample($c$))
7: end for
8: while $H$ is nonempty do
9:   $\beta \leftarrow H$.ExtractMin()
10:  if $\beta \cap \mathcal{P} = \phi$ then
11:    // $\beta$ does not conflict with $\mathcal{N}$.
12:    $\mathcal{N} \leftarrow \{\beta\} \cup \mathcal{N}$
13:    $\mathcal{P} \leftarrow \mathcal{P} \cup \beta$
14:  end if
15:  $c \leftarrow g(\beta)$
16:  $c \leftarrow c \backslash \mathcal{P}$
17:  $H$.Push(Best_Sample($c$))
18: end while
19: return $\mathcal{N}$
```

that generated $\beta$ again. Each time we do the MBFF sampling of a clique, we will select a new target-feasible candidate MBFF which does not conflict with any MBFF belonging to $\mathcal{N}$. We repeat these steps until all the FFs are allocated into some MBFFs of $\mathcal{N}$. Let $n$ be the number of FFs in the original circuit. It is clear that $|\mathcal{N}|$ cannot be greater than $n$. The for-loop of lines 5–7 takes $O(n|\mathcal{C}|)$ time. As for the while-loop of lines 8–18, in the best case it will iterate $O(\mathcal{N})$ times and take $O(|\mathcal{N}|(log|\mathcal{C}| + n))$ time. In the worst case, it will iterate no more than $|\mathcal{N}||\mathcal{C}|$ times and take $O(|\mathcal{N}||\mathcal{C}|(log|\mathcal{C}| + n))$ time.

We use an example to illustrate the procedure of MBFF extraction. We are given a MBFF library that contains 1/2/4-bit MBFFs. There are seven FFs $v_1{\sim}v_7$ and all of them are 1-bit FF. After applying sweep line method, we get two maximal cliques, $c_1 = \{v_1, v_2, v_3, v_6, v_7\}$ and $c_2 = \{v_4, v_5, v_6\}$. Suppose that after the initial sampling, we get MBFF $\{v_1, v_2, v_3, v_6\}$ from $c_1$ and $\{v_4, v_5\}$ from $c_2$, where Cost($\{v_1, v_2, v_3, v_6\}$) < Cost($\{v_4, v_5\}$). In the first iteration, we extract $\{v_1, v_2, v_3, v_6\}$ and put it into $\mathcal{N}$, then generate a new MBFF, said $\{v_7\}$, from $c_1$ after updating $c_1$. The value of each set after this iteration would be $H = \{\{v_4, v_5\}, \{v_7\}\}$, $\mathcal{N} = \{\{v_1, v_2, v_3, v_6\}\}$, $\mathcal{P} = \{v_1, v_2, v_3, v_6\}$, $c_1 = \{v_7\}$ and $c_2 = \{v_4, v_5, v_6\}$. Again, we extract a new MBFF, $\{v_4, v_5\}$, with the smallest cost from $H$. After the second iteration, the sets would be updated as follows: $H = \{\{v_7\}\}$, $\mathcal{N} = \{\{v_1, v_2, v_3, v_6\}, \{v_4, v_5\}\}$, $\mathcal{P} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, $c_1 = \{v_7\}$ and $c_2 = \{\}$. In the last iteration, we extract $\{v_7\}$ from $H$, and get $\mathcal{N} = \{\{v_1, v_2, v_3, v_6\}, \{v_4, v_5\}, \{v_7\}\}$, which is the set of MBFFs ready for location assignment.

## C. Decide MBFF Locations

For a nonconflicting MBFF $\beta$, we first calculate the weighted median interval [26] of the $x$-($y$-)coordinate of its

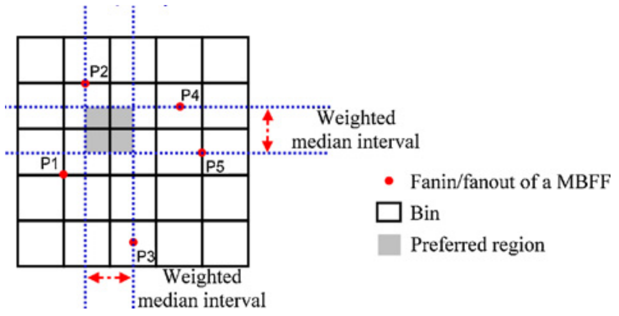

Fig. 9. Example of preferred region. The switching rate ratio of the nets connected to a certain MBFF from P1 to P5 are 2:1:1:3:1.

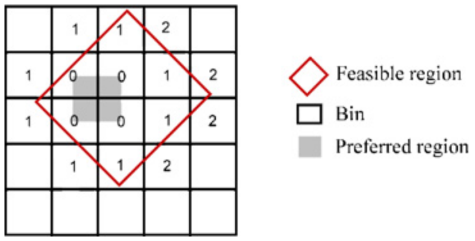

Fig. 10. Ranking of bins.

fanins and fanouts, where the weight of a pin is the switching rate of the signal net between $\beta$ and the pin.

We call the region formed by the intersection of these two weighted median intervals the preferred region of $\beta$, as shown in Fig. 9. We call the bins covered by the preferred region the preferred bins. If we can put MBFF $\beta$ in its preferred region, then the switching power of the signal nets connected to $\beta$, $\sum_{\text{net}_i \in \text{Net}(\beta)} (\alpha_i \times \text{WL}_i)$, will be minimized. We set the rank of preferred bins to 0 and assign increasing ranks to other bins inside $\beta$s feasible region as in Fig. 10. Therefore, each selected MBFF has a ranked-list of bins, and we perform bin assignments according to these ranked-lists, and then do the grid assignments. Nevertheless, we might fail to assign some nonconflicting MBFFs due to placement density constraint. So, if a nonconflicting MBFF generated cannot be successfully assigned, we will generate a smaller MBFF from its originating clique until we can assign it successfully.

The schema of grid point assignment is like bin assignment. For each MBFF, if the bin assignment will not cause bin density violation, then we will find all the preferred grid points. And we run a breath-first search to find an unoccupied grid point to place the MBFF starting from preferred grid points. With this assignment method, we can optimize the switching power of the signal nets connected to the MBFFs.

The time complexity for placing a MBFF $\beta$ is analyzed below. Let $p_\beta$ be the total number of fanins and fanouts, and $b_\beta$ be the number of bins that the feasible region of $\beta$ covers. And let $g$ be the number of grid points in a bin. First, computing the weighted median interval of the $x$-($y$-)coordinate takes $O(p_\beta log(p_\beta))$ time. It takes at most $O(b_\beta)$ time to find a bin inside the feasible region that can accommodate $\beta$ without violating placement density constraint. Then it takes at most $O(g)$ time find an unoccupied grid point in the selected bin. Therefore, the overall time complexity is $O(p_\beta log(p_\beta)+b_\beta+g)$, where the first term is virtually a small constant because $p_\beta$ is no more than 10 in practice.

TABLE II
NUMBER OF FFS, BINS, AND GRID POINTS FOR EACH CASE

| Test Cases | # FFs | # Bins | # Grid Points |
|---|---|---|---|
| t0 | 120 | 6 × 6 | 600 × 600 |
| t1 | 60 000 | 100 × 300 | 2000 × 3000 |
| t2 | 5524 | 100 × 200 | 2000 × 2000 |
| t3 | 953 | 30 × 160 | 600 × 1600 |
| r1 | 267 | 100 × 100 | 70 000 × 70 000 |
| r2 | 598 | 100 × 100 | 94 000 × 95 000 |
| r3 | 862 | 100 × 100 | 99 000 × 97 000 |
| r4 | 1903 | 100 × 100 | 127 000 × 127 000 |
| r5 | 3101 | 100 × 100 | 146 000 × 143 000 |

TABLE III
MBFF LIBRARY FROM [15]

| Bit Number | Normalized Power Consumption Per Bit | Normalized Area Per Bit |
|---|---|---|
| 1 | 1.00 | 1.00 |
| 2 | 0.86 | 0.96 |
| 4 | 0.78 | 0.71 |

## V. EXPERIMENTAL RESULTS

We implemented the algorithm in C++ and conducted our experiments on an Ubuntu workstation with 16 GB memory and two 4-core Intel(R) Xeon(R) E5506 at 2.13 GHz CPUs. The part of the program for finding all maximal cliques is implemented by POSIX threads with eight cores. The rest of the program is implemented by a single thread only.

### *A. Clock Sink Reduction and Signal Net-Switching Power Reduction*

We used benchmarks t0–t3 from [27] and r1–r5 from [28] as our test cases. All original FFs in these benchmarks are 1-bit FFs. The number of bins and number of grid points for each case are shown in Table II. For t0–t3, they provide the locations of FFs and their fanin/fanout pins and the slack to each pin. For r1–r5, the original benchmark files only provide the locations of FFs. So, we carefully assign the location of their fanin and fanout pins such that all FFs are located inside the bounding box of its fanin and fanout pins. It reflects an ideal wirelength-driven placement result.[5] For t0–t3, the original locations of the FFs are not highly optimized. So, we applied the technique in Section IV-C to optimize them to obtain the preoptimized placement results as our baseline. We employed a MBFF library containing 1/2/4-bit MBFFs as shown in Table III.

In the first experiment, we assume the switching rates of signal nets are from 0.05 to 0.15, and $\lambda$ in (2) is set to 0.9 for t0–t3 and 0.5 for r1–r5. (In Section VI-A, we will discuss how the value of $\lambda$ is chosen.) The results are shown in Table IV. Table IV shows that the number of FFs can be reduced by 59%–74%. As expected, the wirelength and switching power of the nets connected to the FFs are bound to increase after FF

[5]In practice, we cannot expect that every FF can be placed inside the bounding box of its fanin and fanout pins.

TABLE IV
REDUCTION OF CLOCK SINKS, WIRELENGTH, AND SWITCHING POWER OF THE NETS CONNECTED TO THE FFs WITH THE RANGE OF SWITCHING RATE FROM 0.05 TO 0.15

| Test | # FFs | | | $\sum_{\text{net}_i \in F} \text{WL}_i$ | | | $\sum_{\text{net}_i \in F} \alpha_i \times \text{WL}_i$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Cases | Preoptimized | Final | Reduction (%) | Preoptimized | Final | Reduction (%) | Preoptimized | Final | Reduction (%) |
| t0 | 120 | 41 | 65.83 | 43 545 | 59 245 | −36.05 | 4603 | 6065 | −31.77 |
| t1 | 60 000 | 15 773 | 73.71 | 24 775 695 | 29 155 135 | −17.68 | 2 477 315 | 2 856 020 | −15.29 |
| t2 | 5524 | 1693 | 69.35 | 1 489 635 | 1 892 755 | −27.06 | 150 014 | 187 039 | −24.68 |
| t3 | 953 | 296 | 68.94 | 258 140 | 349 420 | −35.36 | 26 318 | 34 440 | −30.86 |
| Average | | | 69.46 | | | −29.04 | | | −25.65 |
| r1 | 267 | 109 | 59.18 | 1 679 467 | 1 942 665 | −15.67 | 168 543 | 195 247 | −15.84 |
| r2 | 598 | 237 | 60.37 | 3 721 520 | 4 098 012 | −10.12 | 372 784 | 409 069 | −9.73 |
| r3 | 862 | 316 | 63.34 | 5 201 592 | 5 755 914 | −10.66 | 525 979 | 579 358 | −10.15 |
| r4 | 1903 | 650 | 65.84 | 11 375 578 | 12 505 352 | −9.93 | 1 134 641 | 1 243 214 | −9.57 |
| r5 | 3101 | 1024 | 66.98 | 18 524 756 | 20 373 490 | −9.98 | 1 842 221 | 2 018 665 | −9.58 |
| Average | | | 63.14 | | | −11.27 | | | −10.97 |

TABLE V
RUNTIMES OF ALL STAGES WITH RANGE OF SWITCHING RATE FROM 0.05 TO 0.15

| Test | Runtime (s) | | | |
|---|---|---|---|---|
| Cases | Find Max Cliques | MBFF Extraction | Relocation | Total |
| t0 | 0.01 | 0.04 | 0 | 0.05 |
| t1 | 32.43 | 372.02 | 0.51 | 404.96 |
| t2 | 0.09 | 3.76 | 0.02 | 3.87 |
| t3 | 0.08 | 1.47 | 0.01 | 1.56 |
| r1 | 0.01 | 0.01 | 0.36 | 0.38 |
| r2 | 0 | 0.03 | 1.06 | 1.09 |
| r3 | 0.01 | 0.06 | 1.52 | 1.59 |
| r4 | 0.01 | 0.28 | 5.43 | 5.72 |
| r5 | 0.01 | 0.57 | 10.94 | 11.52 |

merging compared to the preoptimized placements. To control the switching power of the nets connected to the FFs, we can adjust the value of λ in (2). And we will discuss the effect of λ in Section VI-A. The detailed runtimes are shown in Table V. The total runtime is about 45% of our preliminary version presented in [19]. We would also like to bring to notice that in [19], we made a mistake in the runtimes reported for t0–t3 because we accidentally removed all their blockages.

To show the robustness of our algorithm, we performed another experiment by changing the range of switching rate of signal nets to 0.05–0.5. The results and runtimes are shown in Tables VI and VII. Comparing Tables IV and VI, we can see that the reduction of the number of FFs remains stable for different ranges of switching rate.

## B. Experiment with Gated Clock Tree Synthesis

In this section, we compare the gated CTS results with and without performing FF merging and relocation. We applied [29], the best reported work on low-power gated and buffered clock network construction considering both the physical locations of clock sinks and their activity information, to perform gated CTS. Since it requires the activity patterns of each register to be given, we used the same set of benchmarks as in [29], i.e., benchmarks r1–r5 supplemented with activity patterns for each register. The resistance and capacitance values of wires, buffers, clock gates, and FFs are also set to the same values as in [29]. The activity pattern of each sink in the benchmarks was transformed from the instruction stream and

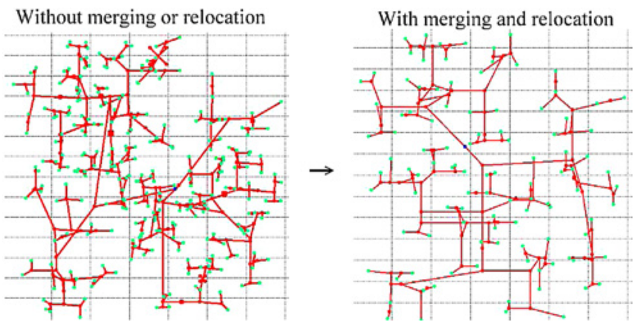

Fig. 11. Gated CTS results of r1 (a) with and (b) without FF merging and relocation.

register-transfer level description of each instruction used in [30]. An activity pattern of a FF is a binary stream composed of "0"s and "1"s, where a "0" means that the FF is idle at that clock period, while a "1" corresponds to an active period of that FF. Switching rates of signal nets connected to a FF are derived from the proportion of the number of "1"s in the activity pattern of each FF. For example, if the activity pattern of FF $\text{FF}_A$ is "0011010100," the switching rates of all the signal nets connected to $\text{FF}_A$ are 0.4. The activity pattern of each MBFF is obtained by performing the bitwise OR operation on the activity patterns of its constituent FFs.

In Table VIII, we report the change in the switched capacitance of the clock network and nets connected to FFs after FF merging and relocation with λ set to 0.5 as in Section V-A. The switched clock capacitance is the sum of the switched capacitances of wires, buffers, clock gates, and FFs on the clock tree, and the enable signal nets. Due to clock gating, the switching rates of enable signal nets and different wire segments, buffers, clock gates, and FFs on the clock tree are different. The switched capacitance of each component is calculated by its capacitance value times its switching rate. Clock Tree WL and Enable Net WL are the wirelengths of gated clock tree and enable signal nets, respectively. # Buffers and # Gates are the number of buffers and clock gates used in the gated clock tree.

It can be seen in Table VIII that the switched clock capacitance is greatly reduced by about 40% in average while the total switched capacitance of the clock network and the FF signal nets is reduced by about 27%. The clock tree WL and

TABLE VI
REDUCTION OF CLOCK SINKS, WIRELENGTH, AND SWITCHING POWER OF THE NETS CONNECTED TO THE FFS WITH THE RANGE OF SWITCHING RATE FROM 0.05 TO 0.50

| Test | # FFs | | | $\sum_{\text{net}_i \in F} \text{WL}_i$ | | | $\sum_{\text{net}_i \in F} \alpha_i \times \text{WL}_i$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Cases | Preoptimized | Final | Reduction (%) | Preoptimized | Final | Reduction (%) | Preoptimized | Final | Reduction (%) |
| t0 | 120 | 39 | 67.50 | 43 545 | 63 055 | −44.80 | 12 100 | 16 977 | −40.30 |
| t1 | 60 000 | 15 737 | 73.77 | 24 775 695 | 29 893 535 | −20.66 | 6 821 728 | 7 837 782 | −14.89 |
| t2 | 5524 | 1701 | 69.21 | 1 489 635 | 1 946 575 | −30.67 | 410 672 | 512 828 | −24.88 |
| t3 | 953 | 296 | 68.94 | 258 140 | 361 590 | −40.08 | 71 816 | 92 861 | −29.30 |
| Average | | | 69.86 | | | −34.05 | | | −27.34 |
| r1 | 267 | 108 | 59.55 | 1 679 467 | 1 950 419 | −16.13 | 460 482 | 526 920 | −14.43 |
| r2 | 598 | 235 | 60.70 | 3 721 520 | 4 127 034 | −10.90 | 1 035 536 | 1 138 761 | −9.97 |
| r3 | 862 | 317 | 63.23 | 5 201 592 | 5 809 658 | −11.69 | 1 392 838 | 1 533 565 | −10.10 |
| r4 | 1903 | 659 | 65.37 | 11 375 578 | 12 616 714 | −10.91 | 3 153 841 | 3 461 954 | −9.77 |
| r5 | 3101 | 1007 | 67.53 | 18 524 756 | 20 556 154 | −10.97 | 5 103 510 | 5 608 925 | −9.90 |
| Average | | | 63.28 | | | −12.12 | | | −10.83 |

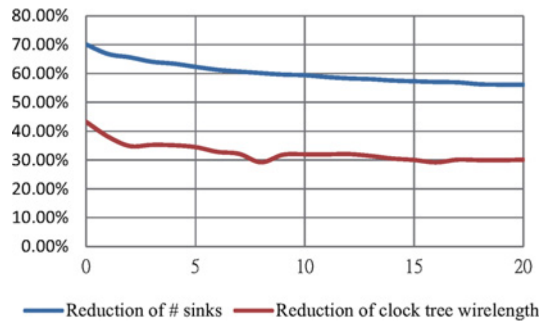

Fig. 12. Reduction of number of sinks and clock tree WL versus the weighting parameter $\lambda$ for r5.

TABLE VII
RUNTIMES OF ALL STAGES WITH RANGE OF SWITCHING RATE FROM 0.05 TO 0.50

| Test | | Runtime (s) | | |
|---|---|---|---|---|
| Cases | Find Max Cliques | MBFF Extraction | Relocation | Total |
| t0 | 0 | 0.05 | 0 | 0.05 |
| t1 | 33.22 | 363.59 | 0.66 | 397.47 |
| t2 | 0.09 | 3.46 | 0.03 | 3.58 |
| t3 | 0.07 | 1.37 | 0.01 | 1.45 |
| r1 | 0.01 | 0.01 | 0.34 | 0.36 |
| r2 | 0 | 0.04 | 1.03 | 1.07 |
| r3 | 0.01 | 0.07 | 1.62 | 1.7 |
| r4 | 0.01 | 0.29 | 6.24 | 6.54 |
| r5 | 0.03 | 0.57 | 11.01 | 11.61 |

enable signal net WL are also greatly reduced by about 37% and 81% in average, respectively. The buffer and gate usages are reduced by about 60% and 85% in average, respectively. The reason is that the smaller the clock tree, the lesser the buffers and clock gates it needs. These results indicate that FF merging is very effective in reducing the switching power, wirelength, buffers, and clock gates of the clock network. Fig. 11 shows the CTS results of case r1 using [29] with and without FF merging and relocation.

We also did some experiments to see the relationship between the reduction of the number of sinks and clock tree wirelength by adjusting the weighting parameter $\lambda$ in (2) as shown in Fig. 12. We can see that the more reduction of the number of sinks the more reduction of clock tree wirelength.

## *C. Modification and Experiment Targeting the Objective of* [15] *and* [16]

References [15] and [16] addressed the problem of merging FFs into MBFFs to optimize the power consumption of FFs only while considering the wirelength of nets connected to FFs as the secondary objective. It assumes that the power consumption of each type of MBFF is constant as in Table III. Our algorithm can be modified to target their objective by adjusting (2) as below to rank the choice of forming MBFF $\beta$ and computing median intervals instead of weighted median intervals in Section IV-C when deciding the MBFF locations as follows:

$$\text{cost}(\beta) = \frac{1}{\text{Pwr}(\beta)} \tag{3}$$

where $\text{Pwr}(\beta)$ is the normalized power per bit of MBFF $\beta$. If the cost of two MBFFs are the same, the one with a smaller value of $(\sum_{\text{net}_i \in \text{Net}(\beta)} \text{EstWL}_i)/(\sum_{\text{net}_i \in \text{Net}(\beta)} \text{OWL}_i)$ will be preferred.

According to the cost function in (3), forming a MBFF with more bits has a smaller cost (see Table III). So we modify our algorithm as follows. In the stage of MBFF extraction, for each clique $c$ we find the maximum MBFF size, $k$, that can be generated and randomly sample $\min(\binom{|c|}{k}, S_{\text{bound}})$ $k$-bit MBFFs from it, then the one with the smallest value of $(\sum_{\text{net}_i \in \text{Net}(\beta)} \text{EstWL}_i)/(\sum_{\text{net}_i \in \text{Net}(\beta)} \text{OWL}_i)$ is chosen as a candidate MBFF.

We compared the results on six test cases from [15] as shown in Table IX.[6] We note that some of the original FFs in these benchmarks are MBFFs. "FF Power Red." is the reduction of power consumed by FFs, and "HPWL Red." is the wirelength reduction of the nets connected to FFs. We can see that our results are very competitive and achieve a good tradeoff between FF power and wirelength.

If we compare the runtimes in Table IX with Tables V and VII, we can see that it is more time consuming to minimize the clock network and switching power of the nets connected

[6]The FF power reduction by [16] is slightly less than reported in their paper because in their paper all original MBFFs in a design were first broken into 1-bit FFs before merging again. But [15] and our work assume that the FFs inside any original MBFF should be kept together, so Table IX reports the results without the preprocessing of breaking original MBFFs into 1-bit FFs.

TABLE VIII
EXPERIMENTAL RESULTS USING [29] TO PERFORM GATED CTS

| Test Cases | # Sinks | | | Switched Capacitance (pF) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Clock Tree | | | Signal Nets to FFs | | | Total | | |
| | Original | Final | Red. (%) | Original | Final | Red. (%) | Original | Final | Red. (%) | Original | Final | Red. (%) |
| r1 | 267 | 106 | 60.30 | 40.36 | 25.75 | 36.19 | 13.42 | 15.27 | −13.80 | 53.78 | 41.03 | 23.71 |
| r2 | 598 | 235 | 60.70 | 85.86 | 54.10 | 36.99 | 29.60 | 32.88 | −11.11 | 115.46 | 86.98 | 24.66 |
| r3 | 862 | 312 | 63.81 | 121.17 | 70.36 | 41.93 | 38.31 | 42.54 | −11.04 | 159.48 | 112.90 | 29.21 |
| r4 | 1903 | 653 | 65.74 | 249.82 | 143.09 | 42.72 | 92.48 | 101.36 | −9.60 | 342.30 | 244.45 | 28.59 |
| r5 | 3101 | 1013 | 67.33 | 377.87 | 218.64 | 42.14 | 137.94 | 151.72 | −9.99 | 515.81 | 370.37 | 28.20 |
| Average | | | 63.58 | | | 40.00 | | | −11.11 | | | 26.87 |
| Test Cases | Clock Tree WL ($\mu$m) | | | Enable Net WL ($\mu$m) | | | # Buffers | | | # Gates | | |
| | Original | Final | Red. (%) | Original | Final | Red. (%) | Original | Final | Red. (%) | Original | Final | Red. (%) |
| r1 | 1 357 515 | 916 882 | 32.46 | 675 915 | 189 123 | 72.02 | 39 | 4 | 89.74 | 41 | 9 | 78.05 |
| r2 | 2 712 849 | 1 808 966 | 33.32 | 1 862 950 | 64 844 | 96.52 | 47 | 19 | 59.57 | 60 | 4 | 93.33 |
| r3 | 3 828 089 | 2 241 628 | 41.44 | 1 973 406 | 258 780 | 86.89 | 48 | 27 | 43.75 | 72 | 8 | 88.89 |
| r4 | 7 394 627 | 4 384 373 | 40.71 | 3 357 815 | 542 775 | 83.84 | 75 | 39 | 48.00 | 92 | 13 | 85.87 |
| r5 | 10 519 743 | 6 457 872 | 38.61 | 4 211 451 | 1 404 037 | 66.66 | 118 | 51 | 59.57 | 128 | 25 | 80.47 |
| Average | | | 37.31 | | | 81.18 | | | 59.57 | | | 85.32 |

TABLE IX
COMPARISON WITH [15] AND [16] AFTER ADAPTING OUR ALGORITHM TO THEIR OBJECTIVE

| Test Cases | # FFs | [15] | | | [16] | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | FF Pwr Red. (%) | WL Red. (%) | Runtime (s) | FF Pwr Red. (%) | WL Red. (%) | Runtime (s) | FF Pwr Red. (%) | WL Red. (%) | Runtime (s) |
| c1 | 98 | 14.83 | 8.7 | 0.01 | 16.69 | −3.57 | 0.01 | 15.64 | 10.13 | 0.02 |
| c2 | 423 | 16.94 | 5.3 | 0.02 | 18.83 | −10.26 | 0.01 | 17.33 | 15.76 | 0.06 |
| c3 | 1692 | 17.07 | 5.2 | 0.08 | 18.93 | −13.35 | 0.03 | 17.47 | 14.76 | 0.19 |
| c4 | 5129 | 16.85 | 5.5 | 0.28 | 18.66 | −12.94 | 0.08 | 17.12 | 14.11 | 0.60 |
| c5 | 10 575 | 17.14 | 5.1 | 0.67 | 18.94 | −13.82 | 0.20 | 17.48 | 14.62 | 1.29 |
| c6 | 169 200 | 17.18 | 5.1 | 116.60 | 18.92 | −13.72 | 4.49 | 17.46 | 14.43 | 22.96 |
| Average | | 16.67 | 5.82 | | 18.50 | −11.28 | | 17.08 | 13.97 | |

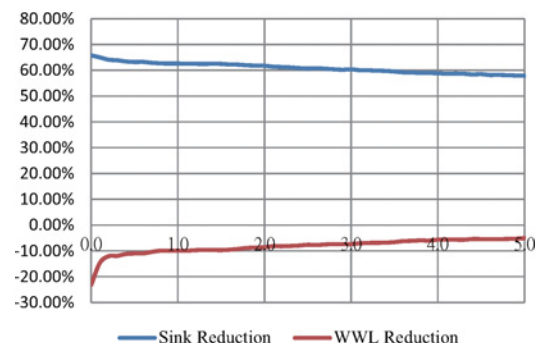

Fig. 13. Average reduction of number of sinks and switching power of the nets connected to FFs versus the weighting parameter $\lambda$ for r1–r5.

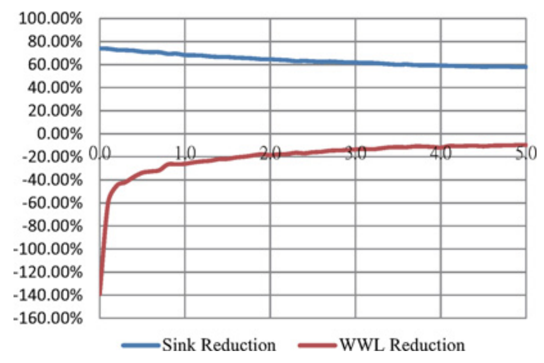

Fig. 14. Average reduction of number of sinks and switching power of the nets connected to FFs versus the weighting parameter $\lambda$ for t0–t3.

to the FFs simultaneously. The reason is that forming a larger MBFF would not necessarily be better because it may greatly increase the switching power of the nets connected to it. So we cannot greedily pick the largest MBFF each time, as a result sampling will take more time and the while loop from line 8 to line 18 in Algorithm 1 will also take more iterations to terminate.

## VI. DISCUSSION

Modern design may include multiple power domains. However, FFs that belong to different power domains cannot be merged together. In our algorithm, it is easy to handle this constraint by performing Procedure 1 for the FFs in each power domain separately. Thus, each maximal clique we found will not contain FFs from different power domains, and they will not be merged at the MBFF extraction stage.

In our algorithm, there are two parameters that affects the quality of results. In Section VI-A, we discuss the effect of $\lambda$ in (2) and try to find a suitable point to increase the overall power reduction. In Section VI-B, we try to find a good sampling bound $S_{\text{bound}}$ in Procedure 2 for high quality of results while limiting the runtime of MBFF extraction.

### *A. Tradeoff Between Reduction of Number of Sinks and Switching Power of Nets Connected to FFs*

We perform an experiment on $\lambda$ in (2) to get the relationship between the number of sinks and switching power of the nets connected to FFs. Figs. 13 and 14 show the average

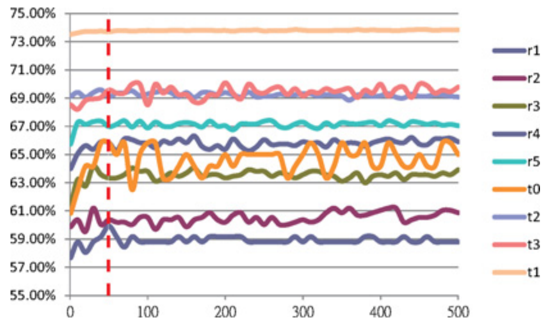

Fig. 15. Reduction of number of sinks versus the sampling bound $S_{\text{bound}}$ for each case.

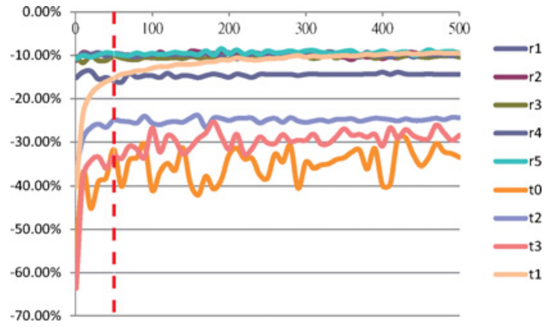

Fig. 16. Reduction of switching power of the nets connected to FFs versus the sampling bound $S_{\text{bound}}$ for each case.

reduction of the number of clock sinks and switching power of the nets connected to FFs versus the parameter λ. Since λ denotes the weight between these two values, a tradeoff situation occurs. It encourages us to find a good balance point in the tradeoff. In fact, the slope of reduction of switching power of the nets connected to FFs is very steep when λ is close to 0 while the slope of reduction of the number of sinks is gentle among all λ. The balance point should be selected according to the target of users. Before FF merging, the user may perform a tentative CTS to get the ratio between the clock network power and switching power of the nets connected to FFs. With this information, the user can decide the value of λ to minimize the overall power consumption in the circuit. Since there are no complete netlists in our test cases, we analyze the average reduction of all test cases in Figs. 13 and 14 and choose $\lambda = 0.5$ for r1–r5 and $\lambda = 0.9$ for t0–t3 to get a good balance between these two targets. In practice, λ should be set design by design as the scales of clock network power and net switching power can vary from design to design.

### B. Sampling Bound $S_{\text{bound}}$

Figs. 15 and 16 show the trend of the reduction of number of clock sinks and the reduction of switching power of the nets connected to FFs when the sampling bound $S_{\text{bound}}$ is varied. The dashed lines is the line for which $S_{\text{bound}}$ is 50. Most curves in Figs. 15 and 16 rise steeply first and become oscillating within a small range soon. The curves for benchmark t0 oscillate within a relatively larger range than other benchmarks due to the fact that t0 contains very few FFs. Based on Figs. 15 and 16, we set $S_{\text{bound}}$ to 50. Note that the value of $S_{\text{bound}}$ could be selected case by case according to some user analysis or time limitation.

## VII. Conclusion

In this paper, we presented a power-driven FF merging and relocation approach to reduce the switching power consumption of a circuit. Our algorithm did a nonconflicting candidate search and determined the location of a MBFF using weighted median interval and ranking. Experimental results showed that the number of clock sinks was reduced by 59%–74%. Moreover, decreasing the number of clock sinks leads to shorter wirelength, lesser buffer and gate usage in the clock tree and hence less clock power consumption. As a result, the switching capacitance of the gated and buffered clock network was reduced by 36%–43%. Finally, the total switching capacitance of clock network and nets connected to the FFs was reduced by 24%–29%. In this paper, controlling the bin placement density was used as a means to maintain routability. As a future work, one may propose a more sophisticated way to address the routability issue.

## Acknowledgment

The authors would like to thank Prof. M. Lin of National Chung Cheng University, Chiayi, Taiwan, for providing them with the executable code and the benchmarks used in [15], and Prof. I. Jiang of National Chiao Tung University, Hsinchu, Taiwan, for providing them with the executable code of [16]. The authors also appreciate the valuable comments of the anonymous reviewers that helped to improve their paper.

## References

[1] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, Jun. 1994.
[2] K. Wang and M. Marek-Sadowska, "Buffer sizing for clock power minimization subject to general skew constraints," in *Proc. Des. Automat. Conf.*, 2004, pp. 153–156.
[3] G. Wilke and R. Reis, "A new clock mesh buffer sizing methodology for skew and power reduction," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Apr. 2008, pp. 227–232.
[4] A. S. Seyedi, S. H. Rasouli, A. Amirabadi, and A. Afzali-Kusha, "Low power low leakage clock gated static pulsed flip-flop," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2006, pp. 3658–3661.
[5] S. Naik and R. Chandel, "Design of a low power flip-flop using CMOS deep sub micron technology," in *Proc. Int. Conf. Recent Trends Inform., Telecommun. Comput.*, 2010, pp. 253–256.
[6] C. C. Yu, "Design of low-power double edge-triggered flip-flop circuit," in *Proc. IEEE Conf. Indust. Electron. Applicat.*, May 2007, pp. 2054–2057.
[7] S. K. Teng and N. Soin, "Low power clock gates optimization for clock tree distribution," in *Proc. Int. Symp. Qual. Electron. Des.*, 2010, pp. 488–492.
[8] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on RTL clock-gating," in *Proc. Des. Automat. Conf.*, 2003, pp. 622–627.
[9] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *IEEE Trans. Circuits Syst. I*, vol. 47, no. 3, pp. 415–420, Mar. 2000.
[10] Y. Cheon, P. H. Ho, A. B. Kahng, S. Reda, and Q. Wang, "Power-aware placement," in *Proc. Des. Automat. Conf.*, 2005, pp. 795–800.
[11] Y. Lua, C. N. Sze, X. Hong, Q. Zhou, Y. Cai, L. Huang, and J. Hu, "Navigating registers in placement for clock network minimization," in *Proc. Des. Automat. Conf.*, 2005, pp. 176–181.
[12] W. Hou, D. Liu, and P. H. Ho, "Automatic register banking for low-power clock trees," in *Proc. Int. Symp. Qual. Electron. Des.*, 2009, pp. 647–652.

[13] R. P. Pokala, R. A. Feretich, and R. W. McGuffin, "Physical synthesis for performance optimization," in *Proc. IEEE Int. ASIC Conf. Exhibit.*, Sep. 1992, pp. 34–37.
[14] Y. Kretchmer, "Using multi-bit register inference to save area and power: The good, the bad, and the ugly," in *EE Times Asia*, 2001.
[15] Y. T. Chang, C. C. Hsu, P. H. Lin, Y. W. Tsai, and S. F. Chen, "Post-placement power optimization with multi-bit flip-flops," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2010, pp. 218–223.
[16] I. H. R. Jiang, C. L. Chang, Y. M. Yang, E. Y. W. Tsai, and L. S. F. Chen, "INTEGRA: Fast multi-bit flip-flop clustering for clock power saving based on interval graphs," in *Proc. Int. Symp. Phys. Des.*, 2011, pp. 115–122.
[17] J.-T. Yan and Z.-W. Chen, "Construction of constrained multi-bit flip-flops for clock power reduction," in *Proc. IEEE Int. Conf. Green Circuits Syst.*, Jun. 2010, pp. 675–678.
[18] Z.-W. Chen and J.-T. Yan, "Routability-driven flip-flop merging process for clock power reduction," in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 2010, pp. 203–208.
[19] S. H. Wang, Y. Y. Liang, T. Y. Kuo, and W. K. Mak, "Power-driven flip-flop merging and relocation," in *Proc. Int. Symp. Phys. Des.*, 2011, pp. 107–114.
[20] G. Chabert and L. Jaulin, "A constraint on the number of distinct vectors with application to localization," presented at the Small Workshop on Interval Methods, Lausanne, Switzerland, 2009.
[21] U. Brenner and J. Vygen, "Legalizing a placement with minimum total movement," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 12, pp. 1597–1613, Dec. 2004.
[22] S. Chou and T. Y. Ho, "OAL: An obstacle-aware legalization in standard cell placement with displacement minimization," in *Proc. IEEE Int. SOC Conf.*, Sep. 2009, pp. 329–332.
[23] H. Imai and T. Asano, "Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane," *J. Algorithms*, vol. 4, no. 4, pp. 310–323, 1983.
[24] M. I. Shamos and D. Hoey, "Geometric intersection problems," in *Proc. IEEE Symp. Foundations Comput. Sci.*, Oct. 1976, pp. 208–215.
[25] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simulat.*, vol. 8, no. 1, pp. 3–30, 1998.
[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms 2/e*. Cambridge, MA: MIT Press, Sep. 2001, p. 194.
[27] *CAD Contest of Taiwan* [Online]. Available: http://cad_contest.ee.ntu.edu.tw/cad10/Problems/B1_Faraday_091223_MultiBitFF.pdf
[28] R. S. Tsay, "Exact zero skew," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, Nov. 1991, pp. 336–339.
[29] W. C. Chao and W. K. Mak, "Low-power gated and buffered clock network construction," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 13, no. 1, pp. 20.1–20.20, Jan. 2008.
[30] J. Oh and M. Pedram, "Gated clock routing for low-power microprocessor design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 6, pp. 715–722, Jun. 2001.

**Shao-Huan Wang** received the B.S. and M.S. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2010 and 2011, respectively.

He is currently an Engineer with Taiwan Semiconductor Manufacturing Company, Hsinchu. His current research interests include very large-scale integrated low-power design and physical design automation.

Mr. Wang and his team earned third place at the 2011 PATMOS Timing Analysis Contest.

**Yu-Yi Liang** received the B.S. and M.S. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2010 and 2011, respectively.

He is currently an Engineer with Taiwan Semiconductor Manufacturing Company, Hsinchu. His current research interests include clock tree synthesis and optimization for high-speed central processing units.

Mr. Liang and his team earned third place at the 2011 PATMOS Timing Analysis Contest.

**Tien-Yu Kuo** received the B.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2010, where he is currently pursuing the M.S. degree.

His current research interests include very large-scale integrated physical design automation and electronic design automation application on graphic processing units.

Mr. Kuo and his team earned third place at the 2011 PATMOS Timing Analysis Contest.

**Wai-Kei Mak** (M'XX) received the B.S. degree from the University of Hong Kong, Pokfulam, Hong Kong, in 1993, and the M.S. and Ph.D. degrees from the University of Texas at Austin, Austin, in 1995 and 1998, respectively, all in computer science.

From 1999 to 2003, he was an Assistant Professor with the Department of Computer Science and Engineering, University of South Florida, Tampa. Since 2003, he has been with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, where he is currently an Associate Professor. His current research interests include very large-scale integrated physical design automation, and field-programmable gate array architecture and computer-aided design.

Dr. Mak supervised a team that earned first place at the 2008 FPT Logic Block Clustering Contest and another team that won the third place at the 2011 PATMOS Timing Analysis Contest. He has served on the Program and/or the Organizing Committee of Asia South Pacific Design Automation Conference, the International Conference on Field Programmable Logic and Applications, and the International Conference on Field-Programmable Technology (FPT). He was the General Chair of FPT in 2008 and was the Technical Program Chair of the same conference in 2006. Since 2009, he has been a Steering Committee Member of the International Conference on Field-Programmable Technology.