Clock Network Optimization with Multi-bit Flip-flop Generation Considering Multi-corner Multi-mode Timing Constraint

Taehee Lee, *Student Member, IEEE,* David Z. Pan, *Fellow, IEEE,* and Joon-Sung Yang, *Member, IEEE*

***Abstract* — Clock network should be optimized to reduce clock power dissipation. The power efficient clock network can be constructed by MBFF (Multi-bit flip-flop) generation and gated clock tree aware flip-flop clumping to pull flip-flops close to the same ICG (Integrated Clock Gating Cell). It is capable of providing an attractive solution to reduce clock power. This paper considers multi-corner and multi-mode (MCMM) timing constraints for the two combined approach. This proposed method is applied to five industrial digital intellectual property (IP) blocks of state-of-the-art mobile SoC (System-on-Chip) fabricated in 14nm CMOS process. Experimental results show that MBFF generation algorithm achieves 22% clock power reduction. Applying a gated clock tree aware flip-flop clumping on top of the MBFF generation further reduces the power to around 32%.**

***Index Terms* — Clock Network Optimization, Clock Gating, Multi-bit Flip-Flop generation, Multi-corner Multi-mode**

I. INTRODUCTION

As today's mobile SoC design becomes complex, the number of flip-flops and clock buffers increases rapidly. Because of the need for low power operation coupled with higher clock frequency in modern high performance designs, clock power becomes one of the most important objectives in mobile SoC designs.

Studies on multi-bit flip-flop (MBFF) generation [1-8], [19] have been proposed to reduce clock power consumption and total flip-flop area by merging multiple flip-flops into a single MBFF. Most approaches present MBFF generations according to the physical position and timing information of individual flip-flops at post-placement stage since there is sufficient physical information of cells available [3-8]. INTEGRA [5] performs an MBFF generation with a fast clustering scheme. It considers the timing information and physical location of individual flip-flops. Compared to [5], [3] proposes a clock gate based MBFF generation which considers not only the timing information of flip-flops but also a distance between ICG and MBFF. The procedure in [3] finds optimal positions of ICGs and flip-flops to minimize the wirelength of clock signal while satisfying clock skew and placement density constraints. However, it does not account for the increase in signal wirelength during the clock gate based MBFF generation. [13] introduces an activity aware flip-flop clumping. It pulls flip-flops with the similar activity pattern into smaller area. However, it does not consider the clock skew, congestion and signal wirelength overhead simultaneously during the flip-flop clumping.

Some IPs in mobile SoCs need to operate under high supply voltage and fast clock speed. On the other hand, the IPs can work in low voltage mode in conjunction with a lowered clock speed to minimize power consumption. Even for the same IPs, they would run at different power modes and various frequencies with dynamic voltage and frequency scaling (DVFS) schemes. In order to reduce the number of timing engineering change order (ECO) which may happen after detailed routing to fix timing violations, multi-corner multi-mode (MCMM) timing constraints should be considered.

The previous works on multi-bit flip-flop (MBFF) generation [1-8], [19] do not consider MCMM during MBFF generation. This paper introduces an integral MBFF generation under MCMM timing constraints in our mobile SoC physical implementation flow (preliminary results were presented in [16]). To further reduce the power consumption in clock network, we apply gated clock tree aware flip-flop clumping, pulling MBFFs toward ICG, since a shorter distance between MBFF and ICG helps to lower the clock power consumption by reducing the number of clock buffers and wire lengths.

The proposed method performs an MBFF generation followed by flip-flop clumping under MCMM conditions. The flip-flop clumping possibly degrades a signal wirelength and clock skew. To avoid the problem, our flip-flop clumping approach considers five constraints: (1) minimum increase in signal wirelength, (2) placement density, (3) clock skew, (4) MCMM timing, and (5) routing congestion constraint. It minimizes a clock power efficiently and prevents the increase in signal wirelength, local congestion, clock skew and MCMM timing violation. As a result, the proposed MCMM clock network optimization approach consisting of (1) MBFF generation and (2) gated clock tree aware flip-flop clumping can minimize clock power efficiently. Fig. 1 briefly illustrates the proposed clock network optimization.

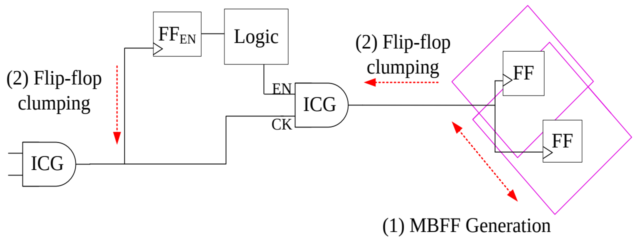


Figure 1. Clock network optimization approach: Multi-bit flip-flop generation and flip-flop clumping.

The important contributions of this paper are summarized as follows:

- The proposed clock network optimization flow considers (1) MBFF generation and (2) gated clock tree aware flip-flop clumping. This method achieves considerable clock power reduction compared to conventional MBFF generation methods.
- The proposed clock network optimization flow is performed under MCMM constraints. It significantly helps to enhance better timing quality of result (QOR) and leads a shorter development cycle by reducing timing ECO iterations.
- A novel cost function for flip-flop clumping is proposed considering clock and logic signal wirelengths and switching activities. Compared to previous flip-flop clumping methods [3], [13], [14], [15], minimum signal wirelength increase, placement density, clock skew, MCMM timing, and routing congestion constraints are considered simultaneously during flip-flop clumping.

To our best knowledge, this is the first work which considers MBFF generation combined with the gated clock tree aware flip-flop clumping under MCMM timing constraints. Compared to a preliminary version of our work [16], this paper has the detailed explanation of MBFF generation and further optimizes a clock network via flip-flop clumping.

The rest of this paper is organized as follows: Section II gives the preliminaries. Section III discusses the proposed MBFF generation under MCMM and Section IV describes the gated clock tree aware flip-flop clumping. The experimental results are given in Section V. The conclusion is in Section VI.

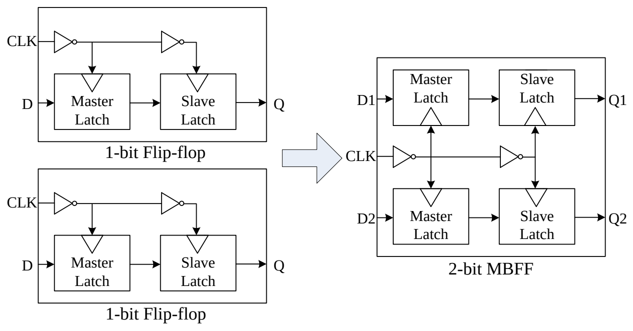


Figure 2. Two single-bit flip-flops merged into one MBFF

II. Preliminaries

In this section, we introduce some preliminaries on MBFF, flip-flop clumping, and MCMM.

A. MBFF

In MBFF, the inverters driving a clock pulse for slave and master latches are shared among the flip-flops. Fig. 2 depicts a 2-bit MBFF structure merging two 1-bit flip-flops. An inverter in each 1-bit flip-flop is able to be shared in a 2-bit MBFF without upsizing in advanced technology [2]. Since the same number of inverters is used in MBFF as a single-bit flip-flop, the normalized power consumption per bit and area per bit of an MBFF cell are consequently smaller compared to multiple single-bit flip-flops. In addition, clock sinks and clock buffers are reduced by MBFF and this lowers clock power consumption.

B. Flip-flop Clumping

Clumping pulls leaf level flip-flops toward ICG and this reduces the net capacitance of clock tree. Because clock nets generally have a high switching activity and the most clock net capacitances are at the leaf level, the flip-flop clumping can significantly save clock power.

C. MCMM

Various operation modes are defined by a design. The mobile SoC operates under different power modes, as well as test modes and functional modes [12]. A corner is defined by the differences due to variations in process, voltage, and temperature. A library corresponding to each corner is provided for MCMM based timing closure. Timing closure for mobile SoC becomes more complex since both multiple modes and multiple corners need to be considered to secure a successful timing closure [17].

III. Proposed MBFF Generation

A. Conventional Approach

For clock power reduction, multiple single-bit flip-flops are merged and form a multi-bit flip-flop (MBFF). In conventional approaches, the MBFF generation is performed considering a physical distance of individual flip-flops and timing information [1-8].

Equation (1) based on an Elmore delay model can be used to translate the timing slack information to its corresponding wirelength [4], [7]. A maximal allowable length, *l*, can be found under the timing constraint, t_{max} by using Elmore delay model.

$$l \le \frac{\sqrt{c_0^2 R^2 + r_0^2 C^2 + 2 r_0 c_0 t_{\max}} - c_0 R - r_0 C}{r_0 c_0} \tag{1}$$

where c_0 and r_0 are a unit capacitance and a unit resistance, respectively. *R* and *C* are a driver strength and a driving load, respectively. Since *l* gives a longest wire length which does not cause timing violations, a *timing slack free region* of each flip-flop is generated using (1). A flip-flop can be placed

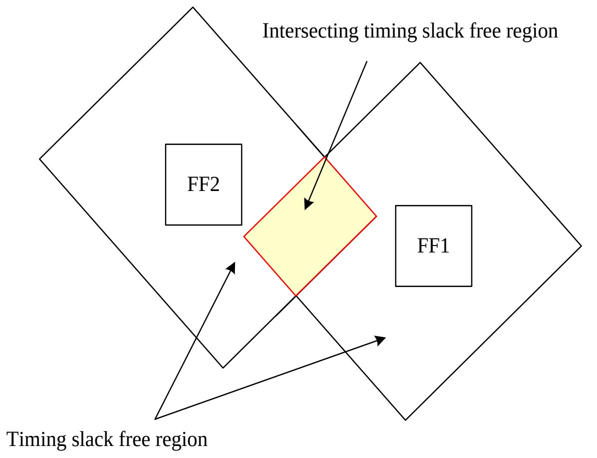


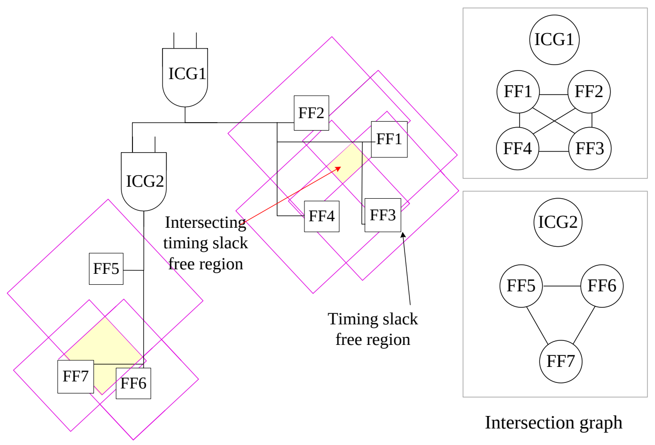
Figure 3. Timing slack free region (Black rectangle for each flip-flop) and intersecting timing slack free region (Red rectangle).

anywhere within the *timing slack free region* such that timing constraint for connected pins of the flip-flop is satisfied as shown in Fig. 3. If there is an intersecting *timing slack free region* among single-bit flip-flops' *timing slack free region*, they can be merged and form an MBFF. The highlighted region in Fig. 3 is the intersection between flip-flop1 (FF1) and flip-flop2 (FF2), hence, they can be replaced by an MBFF.

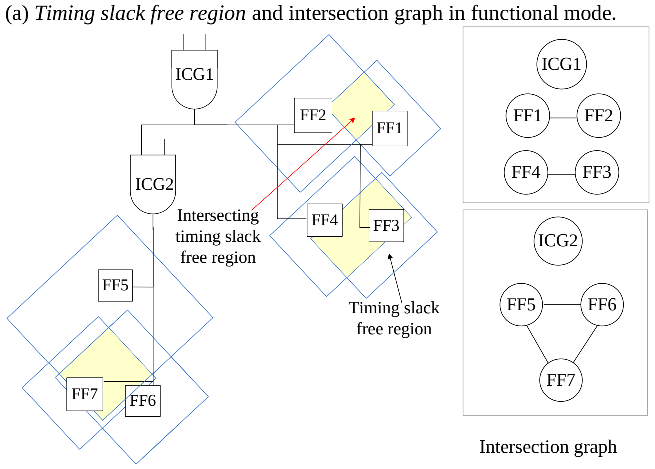
B. Proposed MBFF Generation under MCMM

Modern SoCs dynamically change the operating frequencies depending on working scenarios and environments. If IPs operating with wide-voltage ranges are timing optimized only at either high voltage or low voltage, it may lead timing violations at the other voltage mode. This could happen since a cell delay varies by process, voltage, and temperature (PVT) corners and various modes have different target frequency [9]. Therefore, we consider timing constraints for various modes and corners in MBFF generation. Otherwise, conventional MBFF generation approaches considering a single mode would cause timing violations at other modes or corners. In short, MBFF generation should meet MCMM conditions. Not only should MCMM be considered but ICG also needs to be taken into consideration in this paper. In mobile SoCs, since a clock gating is an important power reduction method for low power [18], ICGs are inserted during logic synthesis and most of flip-flops are driven by ICG. The replacement of single-bit flip-flops by MBFF is performed based on ICGs. We form MBFFs only when the single-bit flip-flops belong to the same ICG. Because they operate at the same clock gating scheme, they can be merged as MBFF. If two flip-flops have different ICGs and their clock gating enable signals could be different, the flip-flops cannot be merged as MBFF.

To perform the proposed MBFF generation flow, a *timing slack free region* is found for each single-bit flip-flop and *intersecting timing slack free region* among them are found. They can be represented as a graph. Each ICG constructs an



(a) *Timing slack free region* and intersection graph in functional mode.



(b) *Timing slack free region* and intersection graph in test mode.

Figure 4. Example of *timing slack free region* and intersection graph for functional mode and test mode

intersection graph, $G(V, E)$, whose vertices (V) represent single-bit flip-flops and edges (E) describe the *intersecting timing slack free regions* between flip-flops. In Fig. 4(a), two intersection graphs, $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, are created. For ICG1, $G_1(V_1, E_1)$ is constructed where $V_1 \in \{ FF1, FF2, FF3, FF4 \}$ and E_1 with $\{ \{FF1, FF2\}, \{FF1, FF3\}, \{FF1, FF4\}, \{FF2, FF3\}, \{FF2, FF4\}, \{FF3, FF4\} \}$. $G_2(V_2, E_2)$ is created for ICG2 where $V_2 \in \{ FF5, FF6, FF7 \}$ and E_2 with $\{ \{FF5, FF6\}, \{FF5, FF7\}, \{FF6, FF7\} \}$.

Note that the proposed method performs MBFF generation under MCMM. Because the *timing slack free regions* and intersecting region of flip-flops in IP would be different depending on various modes such as functional modes and test modes, the intersection graph is subject to each mode. For example, in Fig. 4(a), assume that there is one IP operating at 130MHz in functional mode and the fanouts of FF1, FF2, FF3 and FF4 are declared as a multi-cycle path in timing constraints. Fig. 4(b) shows the same IP in test mode targeting for 100MHz which is slower than functional mode. However, the timing constraints for FF1, FF2, FF3 and FF4 are much tighter than functional mode since they have a single-cycle path in the test mode. The *timing slack free region* of FF1, FF2, FF3 and FF4 in test mode is smaller than functional mode. We then generate intersection graphs for each mode, hence, there are $G_{fm\_1}(V_1, E_1)$, $G_{fm\_2}(V_2, E_2)$, $G_{tm\_1}(V_1, E_1)$, and $G_{tm\_2}(V_2, E_2)$. G_{fm_1} is a graph for functional mode with ICG1, G_{fm_2} is for functional mode with ICG2, G_{tm_1} is a graph for test mode with ICG1, and G_{tm_2} is with test mode under ICG2. The following representations show the vertices and edges for each graph.

$G_{fm\_1}(V_1, E_1)$: $V_1 \in \{ FF1, FF2, FF3, FF4 \}$
$E_1 \in \{ \{FF1, FF2\}, \{FF1, FF3\}, \{FF1, FF4\}, \{FF2, FF3\}, \{FF2, FF4\}, \{FF3, FF4\} \}$
$G_{fm\_2}(V_2, E_2)$: $V_2 \in \{ FF5, FF6, FF7 \}$
$E_2 \in \{ \{FF5, FF6\}, \{FF5, FF7\}, \{FF6, FF7\} \}$
$G_{tm\_1}(V_1, E_1)$: $V_1 \in \{ FF1, FF2, FF3, FF4 \}$
$E_1 \in \{ \{FF1, FF2\}, \{FF3, FF4\} \}$
$G_{tm\_2}(V_2, E_2)$: $V_2 \in \{ FF5, FF6, FF7 \}$
$E_2 \in \{ \{FF5, FF6\}, \{FF5, FF7\}, \{FF6, FF7\} \}$

Once all intersection graphs are generated for ICGs and MCMM, we overlap the graphs under each ICG. For each ICG, there are multiple graphs by MCMM and they are overlapped to find a list of single-bit flip-flop candidates for MBFF. Fig. 5(a) shows the ICG1 and ICG2 graphs generated in Fig. 4 for functional and test modes. The graphs are overlapped and this provides flip-flop candidates for MBFF in Fig. 5(b). This updates the graphs as follows:

$G_1(V_1, E_1)$: $V_1 \in \{ FF1, FF2, FF3, FF4 \}$
$E_1 \in \{ \{FF1, FF2\}, \{FF3, FF4\} \}$
$G_2(V_2, E_2)$: $V_2 \in \{ FF5, FF6, FF7 \}$
$E_2 \in \{ \{FF5, FF6\}, \{FF5, FF7\}, \{FF6, FF7\} \}$

If multiple single bit flip-flops are merged as an MBFF, the signals which are initially connected to single bit flip-flops

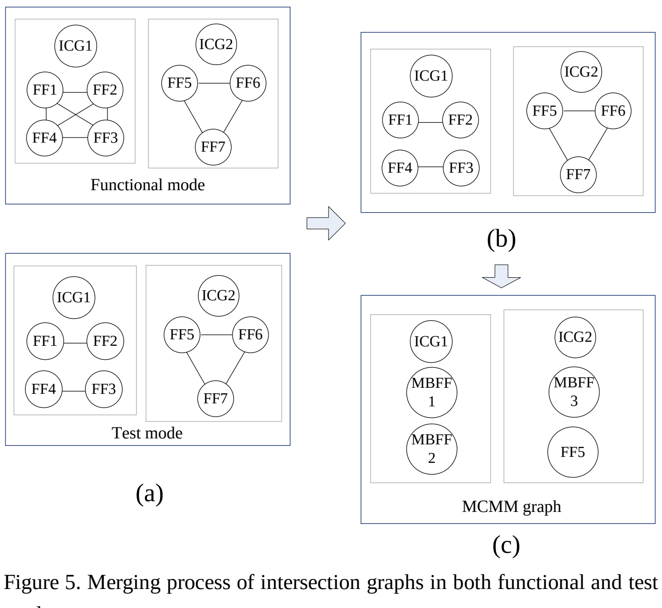


Figure 5. Merging process of intersection graphs in both functional and test mode.

need to be stretched to the MBFF. Hence, the MBFF generation can introduce a signal wirelength increase. To minimize it, the distance between flip-flops needs to be taken into consideration during MBFF generation. The distance between flip-flops is assigned to the edges of the graph in Fig. 5(b).

To generate MBFF, flip-flops are grouped into clusters. It consists of two steps. First, the modified Markov Clustering (MCL) algorithm [20] globally divides flip-flops into multiple clusters depending on the distance (edge weight) between them (Distance aware global clustering). Second, we search maximal cliques within globally divided clusters. (Maximal clique aware local clustering).

MCL groups vertices in a graph based on stochastic matrix operation. Once MCMM aware intersection graph is created for each ICG, MCL generates associate matrix corresponding the MCMM intersection graph. It then conducts two operations on a matrix M: *expand* and *inflate*. The *expand* operation is simple matrix multiplication (MxM). The *inflate* operation emphasizes the inhomogeneity in a column by relatively maximizing large weights on edges and minimizing small weights on edges. MCL performs these two operations iteratively until merging vertices in each cluster.

MCL prefers to cluster edges with high weights, however, the edges with low weights in the graph need to be clustered since the low weight means a close distance between flip-flops. Thus, we divide 1 by the weight to obtain its reciprocal. If the distance between flip-flops is larger than a threshold value, the reciprocal distance is updated by zero in a MCL

Algorithm 1 : MBFF Generation under MCMM

```
Input: a set of ICGs, $S_{ICG}$, a set of flip-flops, $S_{FF}$ driven by each ICG, $I_i$, and a set of scenarios, $S_{scn}$
Output: a set of MBFF, $S_{MBFF}$ driven by each ICG, $I_i$,
 1: For all $scn \in S_{scn}$
 2:   For all $I_i \in S_{ICG}$
 3:     For all $F_j \in S_{FF}$
 4:       find timing slack free region of $F_j$
 5      End For
 6:     For all $F_j \in S_{FF}$
 7:       find intersect timing slack free region of $F_j$
 8:     End For
 9:     create intersection graph of $I_i$
10:   End For
11:End For
12: For all $scn \in S_{scn}$
13:   For all $I_i \in S_{ICG}$
14:     MCMM graph <= merge intersection graph of $I_i$
15:   End For
16: End For
17:For all $I_i \in S_{ICG}$
18:   For all $F_j \in S_{FF}$
19:     generate a set of MBFF, $S_{MBFF}$
20:   End For
21: End For
```

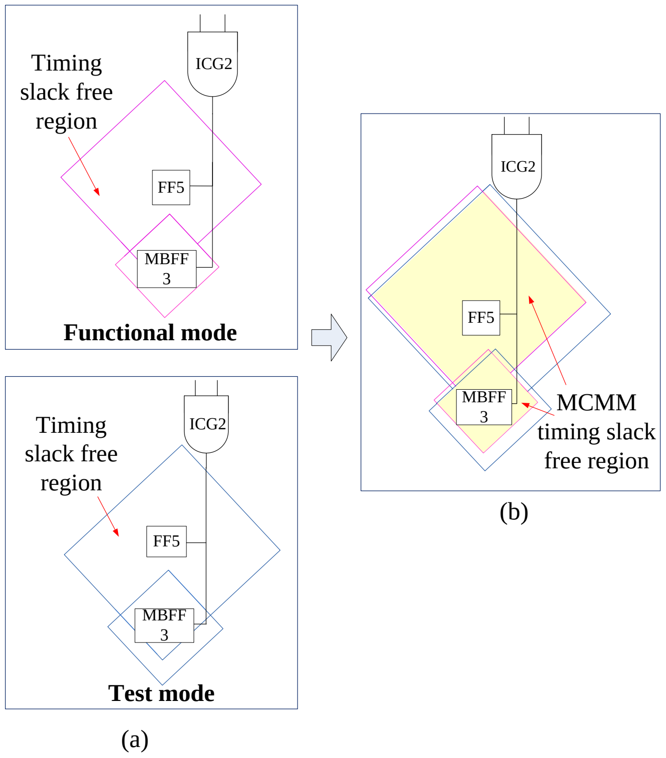


Figure 6. MCMM timing slack free region of flip-flops.

matrix. The zero pruning absolutely reduces chances clustering for flip-flops which are located apart. Hence, the modified MCL does not consider flip-flops whose distance is beyond a threshold distance even though they have overlapping timing slack regions. This helps to find a clique within the threshold distance.

To generate MBFF from single-bit flip-flops, distance aware global clustering and maximal clique aware local clustering are sequentially performed. First, distance aware global clustering globally partitions single bit flip-flops into multiple clusters considering the distance between them. To merge single bit flip-flops, the single bit flip-flops with the low edge values are selected for MBFF generation through MCL based distance aware global clustering. Second, maximal clique aware local clustering explores maximal cliques within globally divided clusters to find a complete subgraph. Flip-flops in a maximal clique are merged into a MBFF.

Fig. 5(c) illustrates an output by the flip-flop clustering algorithm. As can be seen, ICG1 MCMM graph generates two 2-bit MBFFs by merging FF1-FF2 and FF3-FF4. For ICG2, a graph includes three flip-flops. Since a 3-bit MBFF is not available in our MBFF library, two flip-flops are selected to generate one 2-bit MBFF. The distance between FF6 and FF7 is smallest among three flip-flops in the graph, hence, they are chosen to crease a 2-bit MBFF. FF5 is left as a single bit flip-flop. Algorithm 1 describes the procedure to generate MBFF under MCMM.

MCL clustering takes O(n^3). This may look very slow, however, in practice, this does not impose much overhead. For MBFF generation, because the flip-flops under only one ICG are considered, the computational complexity is reasonable in the gated clock network. In addition, the proposed MBFF generation based on MCL clustering can be performed for each ICG in parallel.

There are two main benefits of the MBFF generation based on MCL. First, the existing MBFF generation works [2-5] would make flip-flops move long distance. This induces a signal wirelength increase and the combinational cells are also moved due to the legalization. However, our MBFF generation approach based on MCL clusters flip-flops placed close each other and thus minimizes the disturbance of the original placement. Second, if the timing slack free region of each flip-flop is very large, many flip-flops would be selected as candidates to form MBFFs. In this case, the computation overhead of previous works dramatically increases. However, the proposed approach merges flip-flops though simple matrix multiplications considering physical proximity between flip-flops.

Once MCMM graphs are created, we find *MCMM timing slack free region*. Fig. 6 depicts ICG2, FF5 and MBFF3 illustrated in Fig. 5(c). For FF5 and MBFF3, the size of *timing slack free region* in functional mode would be different from test mode as shown in Fig. 6(a). Their overlapping area between two modes can be identified and this is referred as *MCMM timing slack free region* which satisfies timing constraints for two modes in Fig. 6(b). MBFF is initially placed at the center of *MCMM timing slack free region.*

To reduce clock wirelength, the proposed flip-flop clumping method determines the exact location for MBFF and single-bit flip-flops in *MCMM timing slack free region*, and this will be discussed in the next section.

IV. Gated Clock Tree Aware Flip-Flop Clumping

Once MBFF and *MCMM timing slack free region* are generated, the proposed method performs a gated clock tree aware flip-flop clumping. The flip-flop clumping technique pulls flip-flops (i.e., MBFFs and single-bit flip-flops) close to ICGs in order to further reduce the clock power by reducing a wiring overhead on a clock tree.

During synthesis stage, we limit the number of flip-flops driven by an ICG to the certain user-specified number to prevent timing violations on the enable logic path. Since ICG placement and flip-flop placement significantly influence each other, the ICG location is important. The ICG is placed in the middle of flip-flops driven by it within its timing slack free region that satisfies the enable logic path timing constraint.

The gated clock tree aware flip-flop clumping may cause a logic signal wirelength increase. Assume that there is an 8-bit MBFF. It is connected to eight input and output pins. If the location of the MBFF is moved close to its ICG, input and output signals would be extended in a worst case scenario. This can cancel out the clock power benefit of the flip-flop clumping. To deal with this problem, the proposed flip-flop clumping considers logic signal wirelength increase. In addition to the wirelength increase, we perform the proposed clumping method while satisfying a placement density, clock skew, and routing congestion constraints within *MCMM timing slack free region*.

Firstly, the objective of the proposed flip-flop clumping is to minimize the total clock wire length between leaf flip-flops and ICG. This can be represented as follows:

$$\text{Minimize}: \sum_{i=1}^{m}\sum_{F_j\in I_i}(|XF_j - XI_i| + |YF_j - YI_i|) \tag{2}$$

where a set of ICGs, $S_{ICG} = \{I_1, \cdot\cdot\cdot, I_m\}$ with their location coordinate (XI_i, YI_i) for ICG I_i and a set of flip-flops, $S_{FF}=\{F_1, \cdot\cdot\cdot, F_n\}$ driven by ICG I_i with their location coordinate (XF_j, YF_j) for flip-flop F_j.

If the proposed method pulls flip-flops toward ICGs, the clock wirelength can be reduced. However, as the flip-flops are moving, the logic signals connected to input and output ports of the flip-flops may have to be extended. This can introduce a power consumption increase. Hence, the power reduction by the shortened clock signal length and the power increase by the grown logic signal length need to be examined. To address the trade-off, a switching probability of logic signal and clock net is considered. For example, one flip-flop is moved by clumping, and this reduces the clock signal length and increases the logic signal length. If the logic signal has low switching activities while the clock signal has a high switching probability, a power consumption increase by the lengthened logic signal can be compensated by a power reduction from the shortened clock signal. However, if the logic signal has a high switching activity probability, the clumping will cause more power consumption than before clumping. Before clumping, the logic signal length with its switching probability of one flip-flop can be measured as:

$$SWO_{kj} = \sum_{k=1}^{p}\alpha_k(|XC_k - XFO_j| + |YC_k - YFO_j|) \tag{3}$$

where XFO_j and YFO_j are the original location coordinate for flip-flop F_j. A list of fanin and fanout gates directly connected to F_j is given as $S_{Fanin\_fanout\_gates} = \{C_1, \cdot\cdot\cdot, C_p\}$ with its coordinate (XC_k, YC_k) for C_k when $1 \leq k \leq p$. The total logic signal wirelength is multiplied by its switching probability, α_k to consider the trade-off addressed above. After flip-flop clumping, a total logic signal wirelength of flip-flop F_j with switching probabilities, SWC_{kj}, can also be found as follows:

$$SWC_{kj} = \sum_{k=1}^{p}\alpha_k(|XC_k - XFC_j| + |YC_k - YFC_j|) \tag{4}$$

where XFC_j and YFC_j are the location coordinate for flip-flop F_j after clumping.

In the same manner, the total clock signal wirelength with its switching probability between flip-flop F_j and ICG I_i before (CWO_{ji}) and after (CWC_{ji}) clumping can be represented as:

$$CWO_{ji} = \beta_j(|XFO_j - XI_i| + |YFO_j - YI_i|) \tag{5}$$

$$CWC_{ji} = \beta_j(|XFC_j - XI_i| + |YFC_j - YI_i|) \tag{6}$$

where β_j is a switching probability of the corresponding clock signal to F_j.

Assuming the same unit resistance and capacitance for clock and logic signals, the following equation shows a logic and clock signal wirelength ratio considering their switching activities.

$$\frac{(SWC_{kj} - SWO_{kj})}{(|CWC_{ji} - CWO_{ji}|)} < T \tag{7}$$

(7) implies a power ratio which is a power increase by the logic signal wirelength increase over a clock signal wirelength reduction considering switching probabilities. This should be smaller than T which is a user specified parameter. In our experiments, we set it as 1 for single bit flip-flops and 2-bit MBFFs. For large size of MBFFs (4-bit and 8- bit MBFFs) we set T as less than 1, because we clump large size of MBFF cautiously. Flip-flops can be moved toward ICG while the minimum increase constraint in logic signal wirelength is satisfied not to cause excessive power consumption in signal wirelength.

Secondly, the proposed flip-flop clumping considers a placement density. Since the clumping relocates flip-flops, there is a possibility of impacting the placement density. To deal with the placement density, the chip area is partitioned into a set of bins. Its density can be expressed as:

$$\textit{Density of Bin} = \frac{A_F + A_C}{W * H} \leq D_{\max} \tag{8}$$

where A_F and A_c are an area of all flip-flops and combinational logic cells in a bin, and W and H are a width and height of a bin. This tells how densely cells can be packed in a bin and D_{max} is a design parameter. The clumping would move flip-flops and place them in another bin. This can change the placement density in the bin and may increase the local congestion which possibly leads to a placement legalization failure. To avoid the placement density violation, the proposed method performs a flip-flop clumping only when (8) is still met after clumping.

Thirdly, the proposed method avoids clock skew problems. Since the clumping repositions flip-flops, the clock skew problems may arise. To avoid the problem, we define a maximum allowable skew constraint in (9)

$$|D_l - D_s| \leq S_{\max} \tag{9}$$

where D_s is the least delay between an ICG and the flip-flop, D_l is the largest delay between the ICG and the flip-flop, and S_{max} is a maximum allowable skew.

Lastly, we estimate the routing congestion using fast incremental global routing during MBFF generation and clumping. Before MBFF generation, fast global routing is performed to find routing criticalities at placement stage. Because the full global routing is expensive from a runtime perspective, it cannot be performed frequently. Instead of applying full global routing, we reroute nets between newly generated MBFFs and other cells, and calculate the routing congestion using the previous full global routing information and rerouted wires.

To consider routing congestion by MBFF, routability on the edges of a grid is evaluated. The routing congestion for edges on each side of a grid is defined as the overflow of routing demand over routing capacity of the edge of a grid. Routing demand of the edge consists of 1) global routing demand (the number of global routes along the edge of two adjacent grids) and 2) local routing demand (the local routes within adjacent grids). The number of global routes is captured by global routing. However, since global routing does not consider local routing within a grid, the detail routing information is needed to capture routing resources within a grid. To avoid a runtime overhead, we treat the local routing demand as the number of pins in the grid [21].

$$\frac{RD_{AB} + \gamma \times (n_A + n_B)}{RC_{AB} - RB_{AB}} < P \tag{10}$$

where RD_{AB} is the number of routes determined by global routing on an edge between two adjacent grids, A and B. γ is a pin density factor which is dependent on technology nodes. n_A and n_B are the number of pins in the *Grid A* and *B* respectively. γ x (n_A + n_B) gives the local routing demand within *Grid A* and *B*. RC_{AB} is the total routing capacity cross an edge between grid *A* and *B*. RB_{AB} is routing blockage occupied by fixed cells. The actual routing capacity of an edge is the total routing capacity minus routing blockage. *P* is a user specified parameter. Equation (10) is the ratio between routing demand and routing capacity of an edge, which is considered to be congested if it is larger than *P*. *P* is set as 0.9 in our experiments. It indicates that unless routing demands of four edges of a grid exceed 90% of routing capacity, MBFF cell can be placed.

The proposed method places flip-flops with four constraints discussed above. To perform the flip-flop clumping, we sort the flip-flops in descending size order (i.e, larger bit MBFF to single-bit flip-flop) driven by the same ICG. Because the placement density constraint is more easily influenced by a larger size of flip-flop and the bigger flip-flops have a higher chance of power reduction, the bigger flip-flop (i.e., higher multi-bit flip-flop) is selected first for clumping. If the flip-flops have the same size, one having a longer distance from the ICG moves first than the other having a shorter distance.

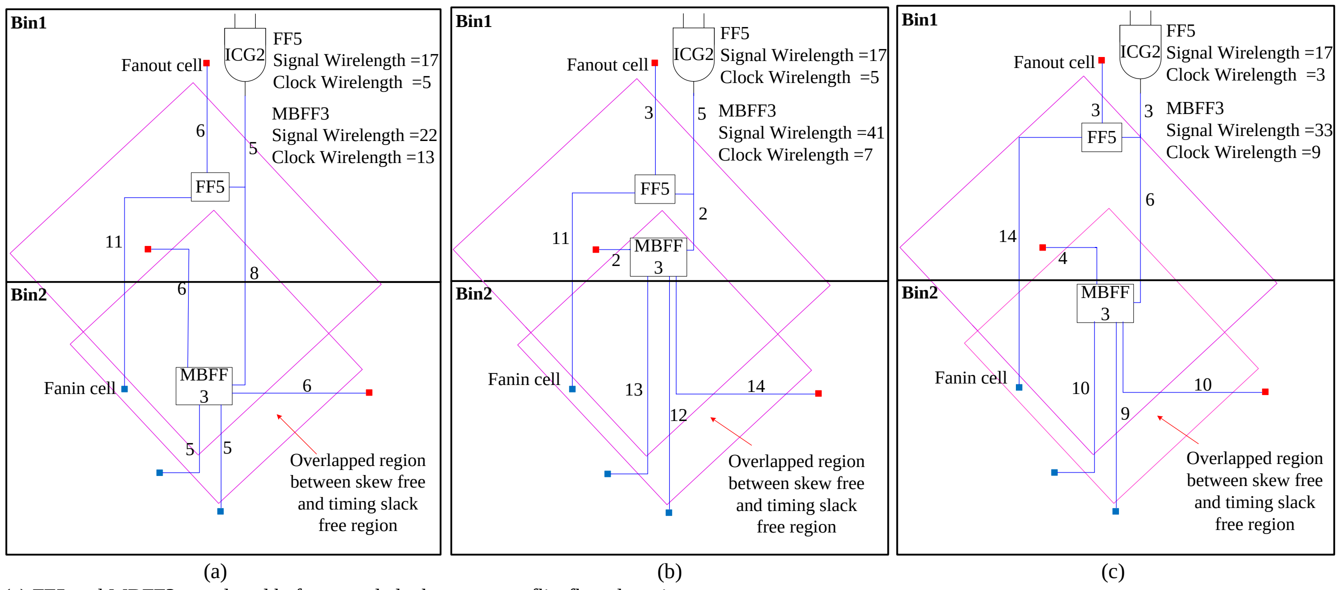
Clock skew free region is created considering the distance between shortest clock path and longest clock path. *Clock skew free region* and *MCMM timing slack free region* overlap to reduce the number of valid grids searched by our algorithm. The proposed method searches a valid grid in a bin within the overlapped region between the *clock skew free* and *MCMM*

Algorithm 2 : Gated Clock Tree Aware Flip-flop Clumping

Input: a set of ICGs, S_{ICG} and a set of flip-flops, S_{FF} driven by each ICG, I_i before clumping

Output: a set of clumped flip-flops

```
1: For all $I_i \in S_{ICG}$
2:   Sort flip-flops in $S_{FF}$ in descending order of size
3:   If flip-flops in $S_{FF}$ have same size
4:     Sort flip-flops in descending order of clock wirelength
5:   End If
6:   For all $F_j \in S_{FF}$
7:     Repeat
8:       Solve Minimize : $|XF_j - XI_i| + |YF_j - YI_i|$ within MCMM timing slack free region
9:     Until (satisfy minimum increase in signal wirelength, placement density, clock skew, and routing congestion constraints described in Equations (7), (8), (9), and (10) )
11:  End For
12: End For
```



(a) FF5 and MBFF3 are placed before gated clock tree aware flip-flop clumping.
(b) MBFF3 does not satisfy a minimum increase constraint in signal wirelength and bin density constraint after gated clock tree aware flip-flop clumping.
(c) Final location for flip-flops satisfying all constraints.
Figure 7. Example of gated clock tree aware flip-flop clumping.

timing slack free region until the density, minimum signal wirelength increase and routing congestion constraints are satisfied. A grid closest to the ICG is first visited and the second closest grid is next searched.

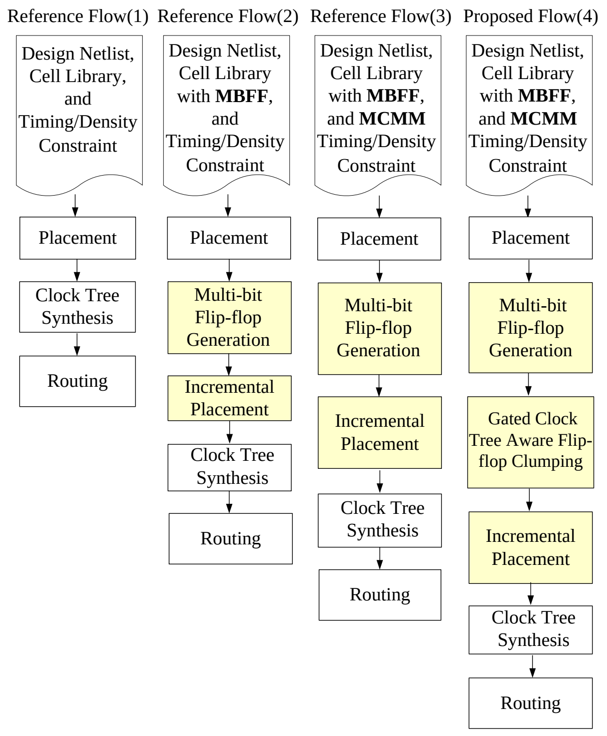
Fig. 7(a) shows an example with flip-flops illustrated in Fig. 6 and their clock and signal wirelengths and locations in each bin before clumping. There are two flip-flops, MBFF3 and FF5, under ICG2. MBFF3 is first selected as a candidate to move toward ICG2 since it has a larger size and longer distance from the ICG2 than FF5. The proposed method first moves it closer to the ICG2 within overlapped region between *MCMM timing slack free and clock skew free region*.

Fig. 7(b) illustrates the location of MBFF3 after clumping. MBFF3 is a 2-bit MBFF and it has two input and output pins. The total signal wirelength is 22 and 41 before and after clumping respectively. The increment in signal wirelength of MBFF3 is 19 (41-22=19) and the decrement in clock wirelength of MBFF3 is 6 (13-7=6). Assuming that four logic signals have the same switching probability and $(\sum_{k=1}^{4}\alpha_k)/\beta$ is 1/3, as given in Equation (7), a ratio between the logic signal wirelength increase over the clock signal wirelength decrease is 19/18 (1/3 * 19/6). This is larger than *T* which is set to 1 in our experiment. It does not satisfy the minimum increase constraint in signal wirelength. Moreover, if the total area of all flip-flops and combinational logic cells in Bin1 exceeds the maximum allowable density constraint, it cannot be placed in Bin1. We move MBFF3 within overlapped region between *MCMM timing slack free and clock skew free region* until all constraints are satisfied.

An adjusted location for MBFF3 is given in Fig. 7(c). The increment in signal wirelength of MBFF3 is 11 (33-22=11) and the decrement in clock wirelength of MBFF3 is 4 (13-9=4). The ratio between increment of signal wirelength and decrement of clock wire length is 11/12 (1/3 * 11/4). It satisfies the minimum increase constraint in signal wirelength. Subsequently, the bin density, clock skew and routing congestion constraints are checked. During the flip-flop clumping, our method checks whether the position of MBFF3 is legalized, because its position should not overlap with other cells. Then, the next candidate, FF5, is selected and moved to the ICG2 within its overlapped region between *MCMM timing slack free and clock skew free region* in the same manner. Fig. 7(c) illustrates the final flip-flops locations by the proposed method. Note that the proposed clumping method can be performed in a parallel fashion for each ICG. Algorithm 2 describes the proposed gated clock tree aware flip-flop clumping algorithm.

V. Experimental Results

For experiments, the five industrial IPs in 14nm technology state-of-the-art mobile SoCs are used. Table I gives target frequencies in MCMM. The target frequencies for test and functional modes are given with operating voltages (0.8V and 0.9V). As Table I shows, the timing constraints are different in MCMM.


Reference Flow(1)
Reference Flow(2)
Reference Flow(3)
Proposed Flow(4)
Design Netlist, Cell Library, and Timing/Density Constraint
Design Netlist, Cell Library with MBFF, and Timing/Density Constraint
Design Netlist, Cell Library with MBFF, and MCMM Timing/Density Constraint
Design Netlist, Cell Library with MBFF, and MCMM Timing/Density Constraint
Placement
Clock Tree Synthesis
Routing
Multi-bit Flip-flop Generation
Incremental Placement
Gated Clock Tree Aware Flip-flop Clumping

(1) Conventional Design Flow without MBFFs.
(2) Design Flow with MBFFs in Single Mode.
(3) Design Flow with MBFFs in MCMM
(4) Proposed Design Flow with MBFFs and Flip-flop Clumping in MCMM
Figure 8. Comparison of four design flows

The proposed method is compared with three other design flows as described in Fig. 8. The following shows details of design flows.

1) Conventional Design Flow : This design flow does not consider MBFF generation and flip-flop clumping. A 0.8V functional mode timing constraint is only considered.
2) Design Flow with MBFF generation in Single Mode : MBFF generation is applied based on INTEGRA [5]. It considers only 0.8V functional mode timing constraint. 2-bit, 4-bit and 8-bit MBFFs are used to replace multiple single-bit flip-flops.
3) Design Flow with MBFF generation in MCMM : MBFF generation is performed with MCMM timing constraints. 2-bit, 4-bit and 8-bit MBFFs are used to replace multiple single-bit flip-flops.

TABLE I. Target Frequency of Five IPs in Each Mode

IP	Target Clock Frequency (MHz)			
	Test mode Operating Voltage = 0.9V : Case 1	Test mode Operating Voltage = 0.8V : Case 2	Functional mode Operating Voltage = 0.9V : Case 3	Functional mode Operating Voltage = 0.8V : Case 4
IP1	460	350	460	350
IP2	500	400	500	400
IP3	600	450	600	450
IP4	650	500	650	500
IP5	650	450	650	450

TABLE II. COMPARISON OF WORST NEGATIVE SLACK (WNS) AND TOTAL NEGATIVE SLACK (TNS) FOR FOUR DESIGN FLOWS

IP	Design Flow	Mode							
		Test mode @ 0.9V		Test mode @ 0.8V		Functional mode @ 0.9V		Functional mode @ 0.8V	
		WNS (ns)	TNS (ns)	WNS (ns)	TNS (ns)	WNS (ns)	TNS (ns)	WNS (ns)	TNS (ns)
IP1	(1)	-0.29	-43.13	-0.23	-29.81	-0.20	-28.47	-0.11	-1.82
	(2)	-0.30	-47.40	-0.24	-32.40	-0.21	-31.29	-0.12	-1.92
	(3)	-0.05	-0.94	-0.08	-1.06	-0.04	-1.32	-0.10	-1.79
	(4) : Proposed	-0.06	-0.84	-0.09	-1.11	-0.06	-1.39	-0.11	-1.87
IP2	(1)	-0.25	-32.65	-0.21	-29.75	-0.06	-1.68	-0.09	-1.80
	(2)	-0.26	-35.88	-0.22	-32.34	-0.06	-1.85	-0.09	-1.89
	(3)	-0.02	-0.40	-0.06	-1.03	-0.01	-0.63	-0.08	-1.78
	(4) : Proposed	-0.02	-0.31	-0.06	-1.02	-0.02	-0.73	-0.10	-2.01
IP3	(1)	-0.31	-49.25	-0.30	-46.52	-0.22	-34.45	-0.11	-3.91
	(2)	-0.33	-54.12	-0.32	-50.56	-0.23	-37.86	-0.12	-4.02
	(3)	-0.06	-1.82	-0.06	-2.19	-0.06	-2.12	-0.10	-4.12
	(4) : Proposed	-0.07	-1.45	-0.05	-2.30	-0.08	-2.25	-0.12	-4.52
IP4	(1)	-0.26	-38.08	-0.25	-35.40	-0.18	-23.06	-0.10	-4.28
	(2)	-0.27	-41.85	-0.26	-38.48	-0.19	-25.34	-0.10	-4.35
	(3)	-0.03	-2.28	-0.06	-2.76	-0.03	-2.01	-0.10	-4.24
	(4) : Proposed	-0.04	-2.51	-0.06	-3.02	-0.09	-2.12	-0.11	-4.60
IP5	(1)	-0.40	-96.62	-0.33	-73.09	-0.34	-77.45	-0.20	-9.50
	(2)	-0.42	-106.18	-0.35	-79.45	-0.36	-85.11	-0.20	-10.00
	(3)	-0.09	-4.95	-0.15	-6.71	-0.03	-4.01	-0.21	-8.23
	(4) : Proposed	-0.10	-5.41	-0.13	-7.21	-0.07	-4.31	-0.19	-9.56

TABLE III. COMPARISONS OF THE NUMBERS OF FLIP-FLOPS, SIGNAL WIRELENGTH, CLOCK WIRELENGTH AND CLOCK BUFFERS FOR FOUR DESIGN FLOWS

IP	Design Flow	# of FFs	# of clock sinks	Signal wirelength × 10^{10} (nm)	Clock wirelength × 10^{8} (nm)	# of clock buffers
		(1-bit / 2-bit / 4-bit / 8-bit)				
IP1	(1)	90542 / 0 / 0 / 0	90542	4.12	48.79	8231
	(2)	814 / 2716 / 5206 / 7934	16670	4.47	32.15	5037
	(3)	2034 / 3076 / 5627 / 7481	18218	4.32	34.80	5352
	(4) : Proposed	2034 / 3076 / 5627 / 7481	18218	4.35	30.53	4515
IP2	(1)	137097 / 0 / 0 / 0	137097	6.23	78.13	12463
	(2)	5133/4690/8494/11706	1379	6.89	53.92	7701
	(3)	2813/4446/8970/11189	27418	6.74	57.24	8175
	(4) : Proposed	2813/4446/8970/11189	27418	6.79	49.84	7121
IP3	(1)	182348 / 0 / 0 / 0	182348	9.21	103.92	15856
	(2)	1640 / 5286 / 10122 / 160206	33254	10.01	66.72	9099
	(3)	4004/6260/10692/15382	36338	9.88	72.08	9851
	(4) : Proposed	4004/6260/10692/15382	36338	9.94	63.70	8766
IP4	(1)	267343 / 0 / 0 / 0	267343	14.22	158.45	23451
	(2)	2670 / 9223 / 16108 / 22725	50726	15.96	112.08	14730
	(3)	7381/10553/17202/21256	56392	15.81	119.40	15690
	(4) : Proposed	7381/10553/17202/21256	56392	15.92	103.90	13648
IP5	(1)	326162 / 0 / 0 / 0	326162	17.91	197.03	29651
	(2)	3268 / 9785 / 20385 / 27723	61161	19.55	137.75	18410
	(3)	8370/11272/21834/25989	67465	19.43	145.49	19442
	(4) : Proposed	8370/11272/21834/25989	67465	19.58	125.67	16613

TABLE IV. COMPARISONS OF POWER AND AREA RATIO FOR FOUR DESIGN FLOWS

IP	Design Flow							
	(1)		(2)		(3)		(4) : Proposed	
	Total Standard Cells Area (um^2)	Dynamic Clock Power (mW)	Area Ratio(%)	Clock Power Ratio(%)	Area Ratio(%)	Clock Power Ratio(%)	Area Ratio(%)	Clock Power Ratio(%)
IP1	362167	52	89	74	90.5	77.0	90.1	67.0
IP2	548388	78	90	76	91.0	78.1	90.6	69.0
IP3	802333	104	88	73	89.8	75.7	89.5	67.5
IP4	1203043	174	92	77	93.7	79.8	93.3	70.3
IP5	1400340	205	90	76	92.2	79.0	91.8	68.1

4) Proposed Design Flow with MBFF generation and flip-flop clumping in MCMM : Both MBFF generation and flip-flop clumping are considered with MCMM timing constraints. 2-bit, 4-bit and 8-bit MBFFs are used to replace multiple single-bit flip-flops.

Table II gives timing comparisons of five IPs with four design flows. The first column shows IPs and the design flows are given in the second column. Timing slacks (WNS : worst negative slack, TNS : total negative slack) are analyzed after clock tree synthesis (CTS) and routing, and given for different modes and operation voltages. The design flow (1) and (2) are optimized for a functional mode with its operating voltage at 0.8V. The design flow (3) and (4) try to optimize the timing in MCMM. This is why WNS and TNS of (3) and (4) are little worse than (1) in 0.8V functional mode (Column 9 and 10). However, as can be seen from other modes in Table II, the MCMM timing constraints consideration achieves considerably better timing quality among most cases. Unless MCMM is considered, more timing ECOs after routing are needed to fix timing related issues. In worst case, we may not be able to fix some timing violations using ECO and this ends up with an IP performance degradation.

Table III lists design details of five IPs by four design flows. The number of clock sinks, the number of single-bit/multiple-bit flip-flops, signal wirelength, clock signal wirelength, and the number of clock buffers are given. The design flow (1) has a single-bit flip-flop and this gives the largest number of clock sinks and the longest clock wirelength. The smallest number of clock sinks is found from the design flow (2) since it generates MBFFs without considering MCMM. Although more number of clock sinks are found in the design flow (3) and (4) than (2), it should be noted that (2) would cause a number of timing violations and need ECO iterations. The MBFF generation from (3) and (4) gives the same results, however, the proposed design flow (4) achieves 11.6% ~ 13.6% shorter clock signal wirelengths by flip-flop clumping than (3). This also gives a clock buffer reduction and considerably helps to reduce clock power. Pulling flip-flops closer to ICGs helps to reduce a clock wirelength, however, this gives 0.7% of a signal wirelength increase from the proposed method than (3).

The power and area comparisons for the four design flows are given in Table IV. The second and third columns give a total standard cell area and dynamic clock power by the design flow (1). The area and the clock power from the other design flows are expresses as a ratio respect to the ones from (1). As can be seen, the MBFF generation helps to reduce an area. In terms of clock power, the proposed design flow (4) achieves the least clock power consumption. Comparing to flow (1), (4) gives almost 32% of the clock power reduction from all IPs. It achieves 7% power reduction comparing to flow (2). The flip-flop clumping pulls flip-flops closer to ICGs and this reduces the number of clock buffers via a clock wirelength reduction. Hence, the proposed method further reduces clock power from the MBFF generation. The experimental results tell that the MBFF generation reduces roughly 22% of clock power comparing to flow (1) and additional 10% clock power reduction is achieved by flip-flop clumping.

If the newly generated MBFF does not find a valid location due to placement density, clock skew, and routing congestion constraints, we disassemble an MBFF to a smaller MBFF or single flip-flops to satisfy all constraints. The number of disassembled flip-flops in IP1 is shown in Table V. Since larger size of MBFFs (8 bit MBFF) are more likely to violate routing or density constraints than smaller size of MBFFs (2 bit MBFF), the more number of large size of MBFFs is disassembled than small size of MBFFs.

In the proposed method, as flip-flops move close to each other, a routability would be degraded. To better demonstrate the contribution of this work, we measure the number of DRC (Design Rule checking) violations of IP1 with design flows after routing as shown in Table VI. Flow (2) causes DRC

TABLE V. NUMBER OF DISASSEMBLED MBFFS

IP	Type of constraints	# of disassembled MBFFs		
		2-bit	4-bit	8-bit
IP1	Density constraint	11	20	25
	Routing constraint	8	24	33
	Clock skew constraint	2	2	3

TABLE VI. DRC VIOLATIONS OF IP1

IP	# of DRC violations				
	(1)	(2)	(3)	(4) : Proposed	Clumping without routing constraint and cost function
IP1	0	89	5	22	128

TABLE VII. COMPARISON OF NORMALIZED RUNTIME OF FIVE IPS FOR THREE DESIGN FLOWS

IP	Design Flows								
	INTEGRA in single mode			Commercial tool in MCMM			Proposed method in MCMM		
	Normalized runtime in placement	Normalized runtime in MBFF generation	Normalized runtime in incremental placement	Normalized runtime in placement	Normalized runtime in MBFF generation	Normalized runtime in incremental placement	Normalized runtime in placement	Normalized runtime in MBFF generation	Normalized runtime in incremental placement
IP1	1	1	1	3.50	3.00	3.36	3.56	3.03	3.32
IP2	1	1	1	2.70	2.38	2.66	2.78	2.44	2.61
IP3	1	1	1	3.37	2.75	3.31	3.41	2.81	3.28
IP4	1	1	1	3.18	2.61	3.03	3.24	2.63	2.96
IP5	1	1	1	3.67	2.97	3.62	3.72	3.05	3.61

violations given since it merges flip-flops without considering distance between flip-flops. This would make flip-flops move long distance to be merged. Our MBFF generation flow (3) merges flip-flops within their near proximity and induces less DRC violations than flow (2). Flow (4), MBFF generation and flip-flop clumping, causes 22 DRC violations. If a routing congestion constraint and a cost function (minimum signal wirelength increase constraint) are not considered for flow (4), the number of DRC violations increases from 22 to 128. It proves that our constraints help to reduce the number of DRC violations in flip-flop clumping.

Our method is incorporated into a commercial physical design tool, Synopsys IC compiler for initial global placement, clock tree synthesis and routing. Owing to restrictions on disclosure of confidential company specific information, we are not at liberty to release much detail information such as runtime. However, to help understanding of runtime information, we provide the runtime for each IP resulting from both commercial tool and our method which is normalized by the result from INTEGRA. As shown in Table VII, the normalized runtimes for five IPs are compared for three design flows: 1) INTEGRA in a single mode, 2) Commercial tool (IC compiler) in MCMM, and 3) the proposed method in MCMM. Only 0.8V functional mode timing constraint is used for INTEGRA in a single mode, however, four different timing constraints (0.8V functional mode, 0.9V functional mode, 0.8V test mode, and 0.9V test mode) are used for commercial tool and our method. Comparing with INTEGRA (Column 2), the commercial tool (Column 5) has 2.7~ 3.67 increase of runtime. It should be noted that INTEGRA does not consider MCMM. The MCMM displays new test mode timing paths that are not visible in a single mode. The MCMM inevitably needs more runtime, since the test mode timing paths require additional effort to optimize them. Compared with the commercial tool, our method results in a relative runtime increase (Column 8 and 9), which is only 1~2% worse than the commercial tool in the initial placement and MBFF generation stage (Column 5 and 6). However, the proposed method archives better timing QOR than INTEGRA and commercial tool since our method considers MCMM timing constraints and prevents a signal wirelength increase. Due to a better timing result and less disturbance of signal data paths, the runtime of our method (Column 10) is less than commercial tool (Column 7) in incremental placement stage.

The proposed method requires additional steps in the design flow. However, the proposed flow can reduce the number of timing ECO iterations after placement and routing and meet the overall tape out schedule successfully since the proposed method considers MCMM for the MBFF generation and flip-flop clumping. Consequently, the proposed method is very effective in reducing clock power and achieving MCMM timing closure.

VI. CONCLUSIONS

In this paper, a clock network optimization approach by the MBFF generation and the gated clock tree aware flip-flop clumping is proposed. The two combined method significantly reduces clock power with a minimum impact on signal wirelength while satisfying MCMM timing constraints. Experimental results with state-of-the-art IPs in 14nm SoCs show that the proposed approach can reduce a clock tree power by around 32% and 7% comparing to conventional design flow and MBFF generation flow in single mode respectively. This paper explains why MCMM needs to be considered in MBFF generation and flip-flop clumping in modern multi-voltage and multi-mode supporting SoC designs. The proposed method is very efficient in clock power reduction and helps to reduce timing ECOs for timing closure.

REFERENCES

[1] C.-C. Tsai, Y. Shi, G. Luo, and I. H.-R. Jiang, “FF-Bond:Multi-bit flip-flop bonding at placement,” in *Proceedings of ACM International Symposium on Physical Design*, 2013, pp. 147–153.

[2] M. P.-H. Lin, C.-C. Hsu, and, Y-C. Chen, “Clock-Tree Aware Multibit Flip-Flop Generation During Placement for Power Optimization," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst*., vol. 34, no. 2, 2015, pp. 280-292.

[3] S-C. Lo, C-C Hsu, and M. P-H Lin, "Power optimization for clock network with clock gate cloning and flip-flop merging" in *Proceedings of International Symposium on Physical Design*, 2014, pp.77-84.

[4] S. H. Wang, Y. Y. Liang, T. Y. Kuo, and W. K. Mak, "Power-driven flip-flop merging and relocation," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, 2012, pp. 180–191.

[5] I. H.-R. Jiang, C.-L. Chang, and Y.-M. Yang, "INTEGRA: Fast multibit flip-flop clustering for clock power saving," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, 2012, pp. 192–204.

[6] S.-Y. Liu, W-T Lo, C.-J. Lee, and H.-M. Chen, "Agglomerative-based flip-flop merging and relocation for signal wirelength and clock tree optimization," *ACM Trans. on Design Automation of Electronic Systems* vol. 18, no. 3, 2013, pp. 40.

[7] 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.

[8] M. P. H. Lin, C. C. Hsu, and Y. T. Chang, "Post-placement power optimization with multi-bit flip-flops," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 30, no. 12, Dec. 2011, pp. 1870–1882.

[9] Y. P. J, Echeverri. M. Meijer, and J. P. D Gyvez, "Logic Synthesis of Low-power ICs with Ultra-wide Voltage and Frequency Scaling," in *Proceedings of IEEE/ACM Design, Automation and Test in Europe Conference,* 2014, pp.1-2.

[10] W. Shen, Y. Cai, X. Hong, and J. Hu, "Activity-aware registers placement for low power gated clock tree construction," in *Proc. IEEE Comput. Soc. Ann. Symp. VLSI*, Mar. 2007, pp. 383–388.

[11] W. Hou, D. Liu, and P. H. Ho, "Automatic register banking for low power clock trees," in *Proc. Int. Symp. Qual. Electron. Dec.*, 2009, pp 647–652.

[12] S. Roy, P. M. Mattheakis, L. M. Navette, and D. Z. Pan, "Clock tree resynthesis for multi-corner multi-mode timing closure," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, 2015, pp. 589-602.

[13] C. Chen, C. Kang, and M. Sarrafzadeh, "Activity-sensitive clock tree construction for low power," in *Proceedings of Int. Symp. on Low Power Electron. Design*, Aug. 2002, pp. 279–282.

[14] 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.*, New Orleans, LA, USA, 2007, pp. 2040–2043.

[15] S. Wimer and I. Koren, "Design Flow for Flip-Flop Grouping in Data-Driven Clock Gating," *IEEE Trans. on Very Large Scale Integration Syst.*, vol. 22, no. 4, 2014, pp. 771-778.

[16] T. Lee, J. Yi, and J. Yang "Multi-bit Flip-flop Generation Considering Multi-corner Multimode Timing Constraint," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2016.

[17] A. Rajaram and D. Z. Pan, "Robust Chip-Level Clock Tree Synthesis," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 30, no. 6, 2011, pp 877-890.

[18] S. Roy, P. Matthaiakis, Pavlos, L. Masse-Navette, and D. Z. Pan, "Evolving Challenges and Techniques for Nanometer SoC Clock Network Synthesis," in *IEEE Int. Conf. on Solid-State and Integrated Circuit Technology (ICSICT)*, Guilin, China, October, 2014, pp1-4.

[19] C. Xu, G. Luo, P. Li, Y. Shi, and I. H.-R. Jiang, "Analytical Clustering Score with Application to Postplacement Register Clustering," *ACM Trans. on Design Automation of Electronic Systems*, vol. 21, no 3. 2, 2016, pp. 41.

[20] S. V. Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, 2000.

[21] Y. Wei, C. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. Reddy, A. D. Huber, G. E. Tellez, D. Keller, and S. S. Sapatnekar, "GLARE: Global and local wiring aware routability evaluation," in *Proceedings of Design Automation Conference,* 2012, pp. 768–773.