# Timing-Driven and Placement-Aware Multi-Bit Register Composition

Ioannis Seitanidis, Giorgos Dimitrakopoulos, Pavlos Mattheakis,
Laurent Masse-Navette, David Chinnery

*Abstract*—**Multi-bit register (MBR) composition is an effective and proven method for clock tree power reduction. The proposed MBR composition follows a balanced restructuring approach that is applied after global or detailed placement. Its goal is to minimize the total number of registers in a design, and simplify subsequent clock tree synthesis, while taking care that any potential degradations in timing slack, wire length, or routing congestion do not offset the power benefits of a lighter clock tree. The proposed methodology identifies nearby compatible registers that can be merged without degrading timing, and without reducing the "useful clock skew" potential. These registers are merged, provided that the MBR placement can be legalized according to the proposed simplified physical constraints. A new integer linear programming formulation minimizes the total number of registers in the design. Additional optimization steps give significant reductions in register count and clock tree capacitance, as shown by experimental results on industrial benchmarks that are already rich in MBRs after logic synthesis. These steps include: MBR decomposition; initial allowance of incomplete MBRs, and the partial recovery of them by the end of the flow; and MBR-specific register sizing.**

*Index Terms*— **Clock tree synthesis, multi-bit registers, physical design, power optimization.**

## I. Introduction

Reduced power consumption is a key design criterion for modern circuits to extend battery lifetime, reduce packaging and cooling costs, and permit higher device performance. Low-power design starts at the architectural level, with techniques such as clock gating that disables the clock signal propagation to the inactive parts of the circuit, and continues through implementation. The challenge in implementation is to create, optimize, and verify the physical layout so that it meets the power budget, as well as timing, performance, and area goals. In this context, clock power optimization is one of the most important objectives, as clock power can contribute 20% to 40% of the dynamic power consumption for a synchronous digital design [1].

The dynamic power consumption is mostly due to switching of capacitances and it is equal to $0.5 \cdot f \cdot C \cdot V_{dd}^2$, for a capacitance $C$ (dis)charging between 0V and supply voltage $V_{dd}$ with switching frequency $f$. Clock gating reduces the switching frequency [2], [3]. Placing registers in clusters [4] reduces the wire capacitance load on the clock network, which can be further reduced by merging them to multi-bit registers [5].

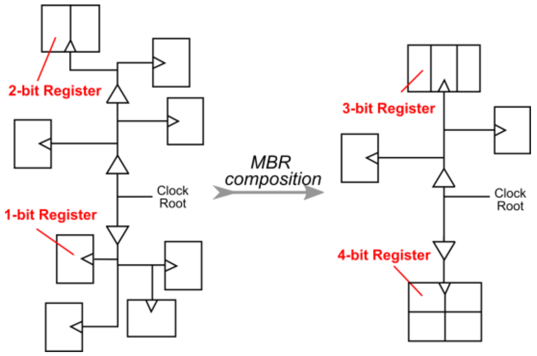


Fig. 1. MBR composition reduces register count and simplifies clock tree synthesis by grouping registers to larger multi-bit cells.

Multi-bit register (MBR) composition reduces the complexity of the clock tree by reducing the number of clock sinks, thus shortening the clock tree's wire length, which decreases the wire capacitance. By sharing clock circuitry within the cell, MBRs also present a smaller pin capacitance load on the clock tree compared to separate single-bit registers. Not only does this reduce the clock switching power at the leaf-level of the tree, but the reduced clock load also allows a smaller clock tree to be used (with fewer and smaller clock buffers) further reducing the clock power. An example of the result of MBR composition is shown in Fig. 1, where the registers of the original design are merged to fewer cells.

MBR composition must carefully select which registers to merge, to maintain the correct function and scan connectivity. It must avoid degrading timing slack, wire length, or routing congestion, while reducing clock power.

The proposed balanced restructuring approach targets MBR composition after global or detailed placement, with the goal to (a) minimize the total number of registers in a design, (b) reduce clock power, and (c) simplify subsequent clock tree synthesis (CTS). The proposed methodology equally applies to circuits that initially have only single-bit registers, or that are rich in MBRs identified earlier in the design flow.

Ioannis Seitanidis and Giorgos Dimitrakopoulos are with the Electrical and Computer Engineering department of the Democritus University of Thrace (DUTH), Xanthi, Greece. (e-mail: iseitani@ee.duth.gr; dimitrak@ee.duth.gr).
Pavlos Mattheakis and Laurent Masse-Navette are with Mentor, a Siemens Business, Grenoble, France. (e-mail: Pavlos_Matthaiakis@mentor.com; Laurent_Masse-Navette@mentor.com)
David Chinnery is with Mentor, a Siemens Business, Fremont USA. (e-mail: David_Chinnery@mentor.com).

### A. Related Work

MBR composition can be performed early in the flow, i.e., during logic synthesis [6–8] for register power reduction. Although early allocation of MBRs offers significant savings, it misses critical placement and timing information that affect the final result. For this reason, the majority of the work in MBR composition is focused on identifying MBRs after global or detailed placement.

In those cases, compatible registers are identified and grouped in MBRs with the goal of minimizing any wire length increase, timing degradation, and/or routing congestion [9–14]. In most cases, the initial designs have only single-bit registers and do not consider any function or library limitations, which are standard restrictions in industrial designs. The composed MBRs are either limited to small sizes of 2 or 4, bits or they move to excessive sizes of up to 16 or 64 bits. In reality, quite a few of the registers may have no logically equivalent multi-bit version, or they may have been specified as fixed or size-only by the designer, and thus cannot be composed to MBRs. The main difference across the various approaches is the clustering or grouping algorithm employed (clique partitioning, analytical or *k*-means clustering, force-driven bonding), and the selection of the placement window within which to search for compatible registers.

MBR composition has been also applied during placement, taking into account the effect of clock tree latency [15]. The late application of MBR composition narrows the design space to identify candidate MBRs. Each new choice requires incremental legalization and clock tree rebuilds, which results in long runtimes and can cause timing hotspots with the disturbance of the clock sink points.

Basic methods for MBR composition have been enriched with other features, such as the optimization of clock gating logic [16], data-driven clock gating [17], and crosstalk avoidance [18]. Recently, MBR composition has been extended to satisfy multi-mode multi-corner timing constraints, where the compatibility of registers is differentiated per mode of operation [19].

Even if registers are not replaced by MBRs their physical clustering can simplify the clock tree and reduce the buffering needed in the clock tree. In these cases, register banks are created in the layout after clustering nearby registers [20–22], with the goal being to create balanced clusters and minimize register displacement from its original position to the new position in the register bank.

### B. Novelties of this MBR composition approach

In this work, MBR composition follows strict rules for identifying compatible registers that can be merged to MBRs. Candidate registers should be compatible in terms of functionality, timing, placement, and scan connectivity. Also, the registers replaced by a MBR should exhibit similar input/output slacks, thus enabling the application of the same useful clock skew after CTS.

To increase the possibility of identifying compatible registers and avoiding any timing incompatibilities, selected MBRs of the original circuit are decomposed and optimized, to facilitate higher quality MBR generation later in the flow.

MBR composition uses a new weighted integer linear programming (ILP) formulation that offers significant reduction in the total number of registers with reasonable runtime. The weights assigned to each MBR candidate correspond to new simplified physical constraints that facilitate MBR detailed placement.

During MBR allocation, we allow incomplete MBRs, where some D/Q pin pairs are left tied-off/disconnected. This reduces register count, while later in the flow some of the unconnected pins are connected using an extra recovery step.

After MBR composition, timing-driven MBR downsizing allows us to save additional clock pin capacitance. This further reduces the clock tree's power consumption.

The combined effect of the new rules that determine register compatibility, the weighted selection of the best MBR candidates, the allowance of incomplete MBRs, and the post-composition optimization steps proposed in this paper, result in significant reductions in register count and clock tree capacitance that lead to simplified and power-efficient clock trees. The presented approach has been integrated into the Mentor® Nitro-SoC™ place-and-route tool, achieving good results in an industrial design flow on modern designs that are already rich in MBRs after logic synthesis.

The rest of the paper is organized as follows. In Section II, we discuss the goals of a successful MBR composition and present briefly the overall MBR composition flow. Section III presents the details of MBR decomposition and optimization. Section IV discusses the compatibility criteria that determine which registers can be composed into MBRs, details the methodology for enumerating MBR candidates and introduces the ILP formulation that minimizes the number of registers. The placement and mapping of the assigned MBRs to specific cells of the library is also discussed in Section IV. Section V introduces the post-MBR composition optimization steps. Section VI presents the experimental results, while conclusions are drawn in Section VII.

## II. Overall Flow And Goals

MBR composition forms MBRs by grouping either single-bit flip flops or latches, or already existing MBRs composed during logic synthesis. The goal is to create larger MBRs, reducing the register count and simplifying the clock tree.

### A. Goals of the MBR composition flow

When two or more registers are selected for merging, they are removed from the netlist and their nets are reconnected to the new MBR. The placement of the new MBR determines the wire length of the reconnected nets, and if not chosen appropriately, may cause timing violations. The candidate registers should have sufficient positive D/Q pin slack to allow them to reconnect to the newly formed MBR without introducing or increasing timing violations.

Any pre-existing MBR, or any newly formed one, should include pins that have similar input D-pin slacks and similar output Q-pin slacks. If the pins of one bit of a MBR have positive D/negative Q slack, and the pins of another bit exhibit negative D/positive Q slack, then those pins contradict possible useful clock skew assignment to the MBR. For example, considering the setup constraints, the pin with negative D slack

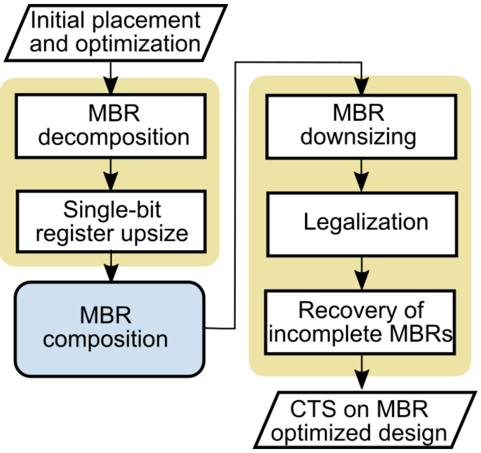

Initial placement and optimization
MBR decomposition
Single-bit register upsize
MBR composition
MBR downsizing
Legalization
Recovery of incomplete MBRs
CTS on MBR optimized design

Fig. 2. The proposed MBR composition flow follows a balanced restructuring approach that reduces the complexity of clock tree synthesis (CTS) without degrading timing, wire-length and routing congestion.

favors a later clock arrival time, while the pin with negative Q slack prefers an earlier arrival [23].

Such cases of timing slack incompatibility should be avoided, either by disallowing candidate registers with incompatible timing profiles to be merged, or by decomposing existing MBRs with such characteristics to smaller MBRs. The pins assigned to each decomposed register (single- or multi-bit) can then be grouped according to their timing slack profile.

Additionally, routing congestion that may arise after MBR composition should not be overlooked. The generation of large MBRs brings many wires in the same region, thus possibly creating routing congestion problems in very dense placements. Given this, the availability of space and wiring resources should not be left as an afterthought, but should be included during MBR selection, mapping, and placement.

### B. The flow for MBR composition

The proposed flow for MBR composition is depicted in Fig. 2. After the initial placement and optimization, MBRs that have bits with positive D/negative Q slack, and other bits with negative D/positive Q slack, are decomposed to smaller MBRs or single-bit registers. Each of the resulting registers should contain bits with the same timing slack profile. In our example, all single-bit registers are upsized, thus increasing the probability of producing more efficient MBR mappings later.

The resulting circuit is then passed to the core of MBR composition. Compatible registers are identified, merged to new MBRs, and appropriately placed, ensuring that the impact on datapath timing, wire length, and routing congestion does not offset benefits of a lighter clock tree.

The composition flow permits the generation of incomplete MBRs, where some D/Q pin pairs are left tied-off/disconnected. Incomplete MBRs tackle the MBR bit-width granularity limitations in typical standard cell libraries, and help reduce register count. We ensure that the merging to incomplete MBRs does not negatively affect the area or leakage power. Although incomplete MBRs are used during MBR composition, they nearly all disappear after a final recovery step at the end of the flow.

Once MBR composition finishes, the MBRs are passed through a sequence of post-processing optimization steps that improve the overall result and simplify the clock tree synthesis that follows. The first step involves MBR downsizing, with the goal of reducing MBR area and clock pin capacitance without degrading timing. The circuit is then legalized to fix any placement violations produced during MBR composition, and redistribute the white space produced by the registers replaced by a MBR. On the legalized circuit, we perform one final optimization step that tries to use as many as possible of the incomplete MBRs' pins by redistributing and reconnecting available nets from nearby registers.

## III. MBR Decomposition And Optimization

Every MBR of the design that contains pins with different timing profiles is decomposed to registers of smaller bit-width (namely, for each MBR where a bit's input-D/output-Q pins have positive D/negative Q slack, and another bit's pins exhibit negative D/positive Q slack). After decomposition, each of the new registers can be either a single-bit register or a MBR, and includes pins with exactly the same timing profile.

During decomposition, we try to minimize the number of decomposed registers. For example, assume the case of the 8-bit MBR shown in Fig. 3, where five pins exhibit positive D/positive Q slack, two pins negative D/positive Q slack, and one pin positive D/negative Q slack. Assuming an example standard-cell library that includes only 1, 2, 4 and 8-bit MBRs, the 8-bit MBR implemented as a two-row cell (according to Fig. 3) is decomposed to (a) one 4-bit MBR and one single-bit register for the five bits with positive D/positive Q slack, (b) a 2-bit MBR for the two bits with negative D/positive Q slack, and (c) one additional single-bit register for the bit with positive D/negative Q slack.

Inside each group (e.g., the group of five pins with positive D/positive Q slack), the separation to MBRs is done according to the available MBRs in the standard cell library and the Q slack of each pin. The pins are sorted according to their Q slack, and then assigned in this order to the largest available MBR of the library. In this way, the pins with large values of Q slack are separated from the pins with less slack. Note that incomplete MBRs are not allowed in this step.

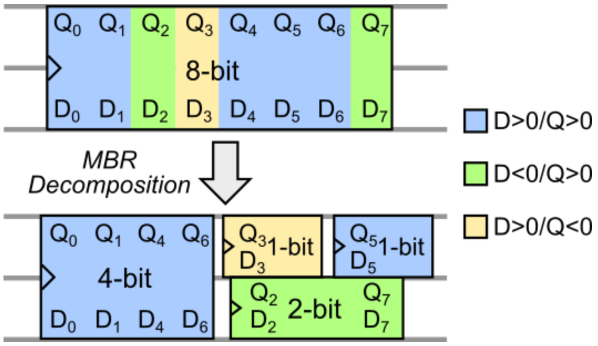


Fig. 3. MBR decomposition of an 8-bit MBR with timing incompatible pins. Decomposition leads to four new registers either MBRs or single-bit registers that are placed in the position of the original MBR. In this example, the 8- and 4-bit MBRs are assumed to be two-row cells.

The new derived cells are placed temporarily at the position of the original MBR, while any useful clock skew properties applied by the designer on the original MBRs are transferred as is to each of the decomposed cells.

After decomposition, the derived cells can be merged with other compatible registers, producing more favorable MBR mappings. The total register count of the design is initially increased by MBR decomposition, but the final number of registers is significantly reduced relative to the original design.

After decomposition, we upsize to maximum size all the single-bit registers with negative Q slack. This improves their output-Q pin timing, which increases the probability of merging with other nearby registers during MBR composition, as detailed in Section IV. This upsizing only minimally affects the timing slack on the input D pin of the register, as verified by the experimental results.

On the other hand, upsizing the single-bit registers increases their clock pin capacitance. This overhead will later disappear, as those registers will likely be replaced by larger MBRs with less total clock pin capacitance than that of the original registers.

MBRs are not upsized at this stage, irrespective of their timing. As verified experimentally, the extra timing benefit that we would earn by upsizing MBRs at this stage does not pay off in reducing either the total register count, or the total clock pin capacitance at the end of the flow.

## IV. MBR Composition

Even if a group of registers has an equivalent MBR in the library to replace them, they cannot be arbitrarily merged to new and larger MBRs. A group of registers can be merged to a larger MBR only if the registers are compatible in terms of functionality, scan chain organization, placement, timing profile, and drive strength.

Once the compatible registers are determined, an ILP-based optimization is formulated that selects which registers should be merged to MBRs. At this step, incomplete MBRs are considered as valid MBR candidates. The weight assigned to each MBR candidate corresponds to new simplified physical constraints that facilitate MBR placement legalization. Once the MBR candidates have been selected, they are mapped to specific library cells and placed, after taking into account the position of the replaced cells and the wire length.

### *A. Compatibility Checks*

For each register in the design, we define a search window that expands the equivalent height of 20 rows in both dimensions. Every register placed inside this search window is checked for compatibility with the current register under examination. Compatibility is checked serially, starting from functional and scan compatibility checks, and continuing with placement, timing, and drive-strength compatibility checks.

Registers can be merged to a new MBR only if there is a MBR in the library with equivalent functionality. For example, a register with a reset pin can only be replaced if a MBR with a reset pin is in the library. Similarly, scan flip-flops can be replaced only if scan-enabled multi-bit versions are available. Quite a few registers may have no logically equivalent multi-bit version, or they may have been specified as fixed or size-only by the designer, and thus cannot be composed to MBRs.

Registers are *functionally compatible* when they share exactly the same control pins, including clock and clock gating conditions. Many papers erroneously assume that any registers in the netlist are functionally compatible, maximizing the opportunities for MBR composition, but this is far from true for real industrial designs. Functional compatibility is performed in two steps, library cell matching and function equivalence. The former checks whether two registers are instantiations of library cells with equivalent library pin functions. For instance, a register with an active low clock enable is marked as incompatible with a register with an active high clock enable, or a register missing clock enable. The function equivalence checks whether the nets connected to each pair of equivalent lib pins are carrying equivalent Boolean functions. For instance, two registers with identical scan enable lib pin functions will be marked as incompatible if the Boolean logic on the nets driving the two scan enable lib pins are not equivalent.

*Scan compatibility* dictates which registers are compatible, based on the scan chain definitions. Registers must be in the same scan partition (i.e., allowed on the same chain). MBRs may have either a single scan-in and scan-out pin, or multiple independent scan in/out pins (the scan enable pin is still shared). In the first case, if the scan pins belong to the same scan partition, then moving scan pins across different scan chains is allowed, and no additional constraints are imposed because of the scan chain definitions. However, registers that belong to ordered scan chain sections may only be composed to a single MBR with an internal scan chain that preserves the same scan order within the MBR. In the second case, where MBR cells with separate scan pins per D/Q pair are used for composition, no restrictions are imposed, as several scan chains with different constraints can cross the same MBR, providing they have a common scan enable signal. Scan compatibility in the absence of separate scan pins per D/Q pair is performed in two steps. In the first step, registers that belong to different scan partitions are marked incompatible. In the second step, registers that belong to the same scan partition are marked as incompatible if they are ordered, but belong to different ordered sections.

In the following steps, registers are checked for *placement compatibility*. For each register, a *timing-feasible placement region* is identified by transforming the positive timing slack of the input D and output Q pins to an equivalent distance that it can move without causing a timing violation.

Each register input (output) slack value defines a diamond. At the center of the diamond is the fanin (fanout) gate and its half diagonal is the equivalent distance. An example of the timing feasible region of a register is shown in Fig. 4(a). We used Elmore delay for the timing slack to equivalent distance calculation similar to [19, 24].

Registers are compatible with respect to placement if their timing feasible regions overlap, as shown in Fig. 4(b). The placement compatibility is checked on a global or detail placed design to give a realistic sense of the relative placement of the registers under consideration for merging.

If the timing slack is negative, the feasible region is limited to the intersection of the bounding boxes of the violating pins with

The final version of record is available at http://dx.doi.org/10.1109/TCAD.2018.2852740

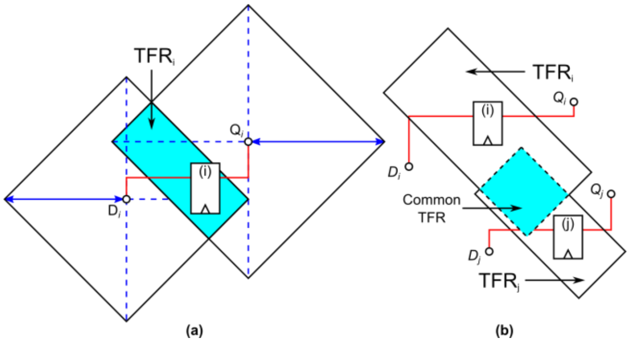


Fig. 4. (a) An example of the formulation of timing feasible region (TFR) of a single bit register. (b) Two registers whose TFRs overlap can be considered placement compatible.

the feasible regions of the rest of the D and Q pins of the same cell. Even if a negative slack does not permit the cell to move, it is not left out of placement compatibility checking, as it has a timing feasible region that matches its footprint, to which other registers with positive slack can possibly move.

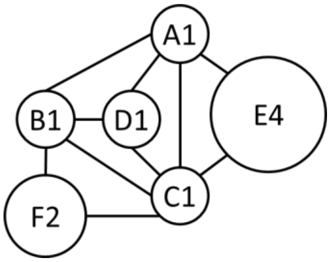
Next, *timing compatibility* is checked, to avoid merging cells that have positive D/negative Q slack with cells that exhibit negative D/positive Q slack. At this point, due to MBR decomposition, there is no MBR with such contradictory timing slacks. With this additional check, we ensure that we don't create new MBRs with this undesirable characteristic.

Even if the D/Q slack signs of two cells are the same, timing compatibility is preserved only if the magnitude of the observed slacks is similar. We should not merge registers with a large difference in timing criticality, because it increases power when a timing-critical signal forces the MBR to be upsized unnecessarily for the other signals. We must avoid very different clock useful skew values, as only one can be realized for a given MBR, and the difference degrades useful skew opportunities for other timing paths to/from the MBR.

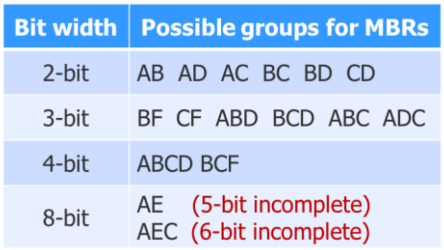
Finally, the last check is *drive-strength compatibility*. Two or more registers are considered compatible if their drive resistance differs by less than 3%. For drive resistance, we refer to a linear model approximation of the register's delay as drive resistance multiplied by load capacitance, with some additional fixed "intrinsic" delay in the register. A cell with low drive resistance can drive more capacitance with less delay. In practice, we use accurate CCS timing models.

We need to avoid merging a high drive strength cell with a low drive strength cell. If this happens, then the derived MBR should be of high drive strength in order not to degrade timing (implicitly the low drive strength register is upsized when merged in the new MBR). However, this would significantly increase the MBR's area and power. By characterizing two registers with different drive strengths as incompatible, we avoid such inefficient outcomes.

The only case in which registers are considered compatible, even if their drive strengths differ, is when a high drive strength register is not timing critical (it has a lot of output-Q pin slack). In this case, when multiple registers are merged to a new MBR, the MBR can use the lower drive strength of the registers from which it came. The high drive strength registers that participate in the new MBR are implicitly downsized, thus relinquishing some of their available slack.



(a)



| Bit width | Possible groups for MBRs |
|---|---|
| 2-bit | AB AD AC BC BD CD |
| 3-bit | BF CF ABD BCD ABC ADC |
| 4-bit | ABCD BCF |
| 8-bit | AE (5-bit incomplete)<br>AEC (6-bit incomplete) |

(b)

Fig. 5. (a) A compatibility graph of six registers, comprising ten bits in total. The compatible registers (nodes of the graph) are connected with an edge. Each register has a name and a size: A1 is a single-bit register; F2 is a 2-bit register; E4 is a 4-bit MBR inserted during logic synthesis. (b) The possible groups for MBRs after clique enumeration.

### B. *MBR Candidate Enumeration and Incomplete MBRs*

The compatible registers of the design are represented by the compatibility graph *G* in Fig. 5(a). The graph nodes are the registers, whether single bit or pre-existing MBRs, and the edges of *G* reflect the compatibility between them.

A MBR can only be formed from registers that are all compatible with each other. Therefore, the registers that can be merged to a new MBR form a clique in *G*. For instance, the 4-node clique {A, B, C, D} and the 3-node clique {B, C, F} can each be mapped to a 4-bit MBR. By enumerating all the cliques of *G*, we determine the set of candidate MBRs to consider during MBR composition.

During clique enumeration, a clique is considered valid if the number of register bits matches the size of at least one MBR in the cell library. For example, for a cell library that consists of 1, 2, 3, 4, and 8-bit MBRs, the 3-node clique {A, C, E} that involves 6 register bits is invalid, since a 6-bit MBR is not available in the library. The clique {A, C, E} is valid if it is allowed to map to an 8-bit MBR, which would be incomplete as only 6 out of the 8 D/Q bits are connected.

The table in Fig. 5(b) lists all cliques for the compatibility graph of Fig. 5(a), and the different bit widths of MBR cells that can be used for their mapping. Cliques {A, E} and {A, E, C} need 5 or 6-bit MBRs that are not available, but they can be mapped to an 8-bit MBR, leaving some pins unconnected.

Incomplete MBRs may seem a waste of area and leakage power, but it can be advantageous as MBRs share the register control pins and associated logic. For example, replacing 7 single-bit registers (with 7 reset and 7 clock connections) with an 8-bit MBR that uses one reset and one clock wire saves 12

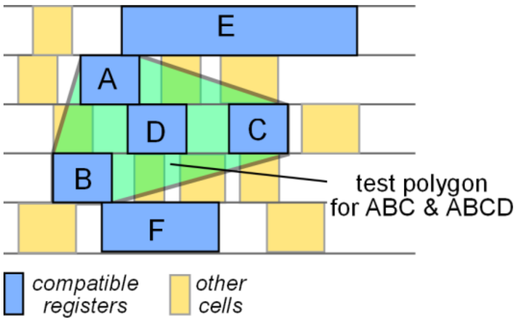


Fig. 6. The initial placement of registers of the compatibility graph in Fig. 5(a). Each register's size corresponds roughly to its bit width (number of D and Q pins). To improve routability after mapping to MBRs, we check inside the polygon surrounding the clique for other registers. The fewer intervening registers, the more favourable the candidate MBR.

wire segments, even if one D/Q pin pair out of 8 is disconnected. However, MBRs with internal scan may not be suitable for this – at the least, the first bit scan input pin and the last bit scan output pin must be connected to a scan chain.

Allowing incomplete MBR cells gives additional freedom to the MBR composition to minimize the number of registers. To keep the area and leakage overhead under control, we only consider an incomplete MBR as a valid candidate for MBR composition when the area of the incomplete MBR does not exceed the area of the replaced registers multiplied by a selected overhead allowance factor. Even if incomplete MBRs are used at this stage, the majority are fully utilized at the end of the flow by reconnecting nets of nearby registers to the empty pins of the incomplete MBRs.

To enumerate all cliques of *G*, we first enumerate all maximal cliques of *G* using the Bron-Kerbosch algorithm [25]. For each maximal clique, we enumerate all the valid sub-cliques for the permitted bit widths per the MBR library cells, using a dynamic programming approach.

The runtime complexity of maximal clique enumeration is O($3^{n/3}$). This is not computationally tractable for large graphs. Hence, *G* is partitioned to a set of connected components, which are further decomposed to a set of sub-graphs using *k*-means clustering. Each sub-graph cannot exceed 30 nodes. The partitioning is driven by the register clock pin positions, using Manhattan distance, to maximize the clock tree power reduction achieved by MBR composition. At each iteration of *k*-means, flops are assigned to clusters, and cluster centers are then recalculated to reflect the new flop-to-cluster assignment. At each iteration, when a cluster reaches its limit of 30 nodes, it cannot accept any new nodes and the unassigned flops at each iteration are assigned to other clusters that are not full. Increasing this bound above 30 did not help, as the slight improvement cost too much additional runtime. As detailed in the experimental results presented in Section VI, the majority of the connected components consist of fewer than 10 nodes and do not require further partitioning. Partitioning is only applied to 5% of the connected components that consist of more than 30 nodes.

### C. ILP Formulation

Clique enumeration defines the set M = {$M_0$, $M_1$, ..., $M_k$} of valid MBR candidates. A register of the original design may participate in various MBR candidates. This attribute is declared via binary variables $a_{ij} \in \{0,1\}$, where $a_{ij} = 1$ if cell *j* participates to MBR candidate $M_i$, otherwise $a_{ij} = 0$. To identify which candidate MBRs are selected from among the MBR candidates, we add a binary variable $x_i \in \{0, 1\}$; $x_i = 1$ when MBR $M_i$ is assigned to replace the constituent compatible registers, else $x_i = 0$. When the register *j* is grouped in MBR $M_i$, and the corresponding MBR is selected, then both $x_i$ and $a_{ij}$ should equal one. The total number of registers is minimized by solving the following integer linear program

$$\textbf{Minimize} \quad \sum_{i=1}^{|M|} w_i x_i$$
$$\textbf{Subject to} \quad \forall register\ j: \sum_{i=1}^{|M|} a_{ij} x_i = 1;\ a_{ij}, x_i \in \{0,1\}$$

The constraint added for each register *j* guarantees that each register will be part of only one MBR. The cost function of the ILP does not treat all MBR candidates equally. Each candidate MBR $M_i$ is associated with a weight $w_i$ ; the smaller the weight $w_i$ , the more favorable the choice of $M_i$.

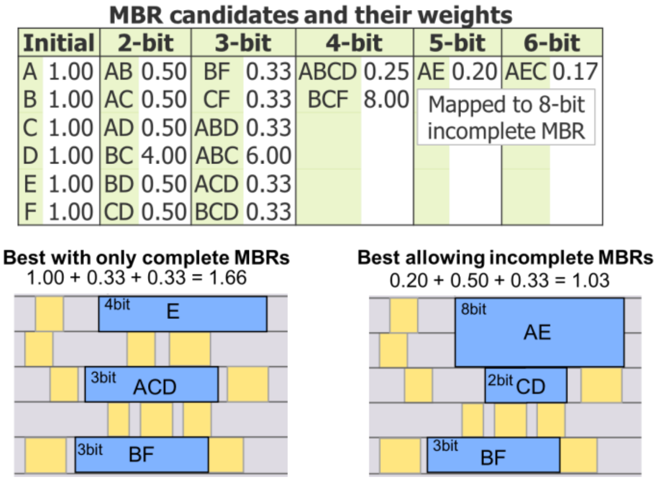
### D. Weights to limit wire-length and congestion

MBRs, due to their multiple input and output pins, lead to wire concentration, increasing the possibility of local routing overutilization. To avoid this, we aim to spread the routing demand to nearby regions by penalizing (with appropriate weights) the composition of new large MBRs very close to other already formed MBRs. In this way, we implicitly handle the possible increase in routing congestion after MBR composition. Considering routing congestion explicitly in the ILP would require the addition of a routing utilization model or more constraints. However, our experimental results in Section VI show that the weighting heuristic chosen to handle the MBR-specific routing demand is adequate for achieving our goal without increasing the initial routing congestion.

The weight assigned to each candidate MBR is based on the relative placement of the compatible registers that will be merged to this MBR. *The most favourable MBR candidate*, which receives the lowest weight, *is one that avoids any other closely-placed compatible registers* that do not participate in the clique and could belong in another composed MBR. This limits the probability that the nets of the two new MBRs cross each other, keeping routing utilization under control.

For each MBR candidate, we define a polygon formed by the corners of the participating registers. We compute the convex hull formed by the outer corners of those registers.

Fig. 6 illustrates the test polygon that corresponds to the 4-node clique {A, B, C, D} or the 3-node clique {A, B, C}, which produce respectively a 4-bit and a 3-bit MBR candidate. All registers of the {A, B, C, D} clique are part of the test polygon, and no other compatible register lies in the same region. So this choice is the most favourable, and receives the minimum weight. The 4-bit MBR candidate has a clear area to be placed physically separate from any other MBR. The empty space, which will be available after removing the participating compatible cells, roughly defines the room to place the MBR. Even though this white space is not contiguous as required to place the MBR, placement legalization is simplified because no other register will be placed in the same area. It also reduces the displacement of non-register cells that exist in the same area. Registers are larger and often have higher placement priority,



MBR candidates and their weights

| Initial | | 2-bit | | 3-bit | | 4-bit | | 5-bit | | 6-bit | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 1.00 | AB | 0.50 | BF | 0.33 | ABCD | 0.25 | AE | 0.20 | AEC | 0.17 |
| B | 1.00 | AC | 0.50 | CF | 0.33 | BCF | 8.00 | Mapped to 8-bit incomplete MBR | | | |
| C | 1.00 | AD | 0.50 | ABD | 0.33 | | | | | | |
| D | 1.00 | BC | 4.00 | ABC | 6.00 | | | | | | |
| E | 1.00 | BD | 0.50 | ACD | 0.33 | | | | | | |
| F | 1.00 | CD | 0.50 | BCD | 0.33 | | | | | | |

Fig. 7. The weights of candidate MBRs for the example in Fig. 5 and the best solution. The 5-bit and 6-bit can map only to an 8-bit incomplete MBR.

so smaller movement of fewer registers helps minimize the placement disturbance.

For the candidate 3-bit MBR of clique {A, B, C} in Fig. 6, the polygon defined by the corners of A, B, and C, includes the compatible register D. This composition is worse, as register D may merge with another MBR that will be placed near the 3-bit MBR, which will increase the routing resource utilization locally.

By weighting each candidate MBR M$_i$, we promote the composition of large MBRs, when the region defined by the constituent compatible registers is clean of other registers. When there are many intervening registers, we promote the selection of smaller, but clean, MBRs. This is achieved with a heuristic weight $w_i$ for each candidate MBR M$_i$ as follows:

$$w_i = \begin{cases} 1/b_i, & n_i = 0 \\ b_i 2^{n_i}, & 0 < n_i < b_i \\ \infty, & n_i \geq b_i \end{cases}$$

$b_i$ is the number of bits of the registers that will be merged to MBR M$_i$, and $n_i$ is the number of other compatible registers that block the convex polygon defined by the outermost corners of the registers replaced by M$_i$. *To favour merging of registers, a weight of 1 is assigned to existing registers.*

A register is a blocking register for M$_i$ if its centre is inside the corresponding test polygon and it is not a constituent register of M$_i$. For the example shown in Fig. 5(a), the clique {A, B, D} has $\{b_i, n_i\} = \{3,0\} \Rightarrow w_i = 1/3$, as it is not blocked by any other register in Fig. 6. Whereas the clique {A, B, C} has $\{b_i , n_i\} = \{3, 1\} \Rightarrow w_i = 6$ because the centre of D is inside the polygon defined by the outmost corners of {A, B, C}.

When the test polygon for each candidate M$_i$ is free of any other compatible registers, the weight promotes the selection of larger MBRs. For instance, the weight of a clean 8-bit MBR is 1/8, which is less than the 1/2 = 1/4 +1/4 weight of two clean 4-bit MBRs needed to cover the same number of bits.

When there are obstacle registers, the selection of large MBRs is penalized relative to the selection of smaller MBRs. Consider an example 8-bit MBR candidate that has one obstacle register, so $\{b_i, n_i\} = \{8, 1\}$. In this case, the weight of this candidate would be $w_i = 16$. The equivalent choice with two smaller 4-bit MBRs would have one clean MBR with $\{b_i, n_i\} = \{4,0\} \Rightarrow w_i = 1/4$, and another 4-bit MBR that includes the intervening register, $\{b_j, n_j\} = \{4,1\} \Rightarrow w_j = 8$. The total cost of the second option is 8.25, causing the ILP to select the two 4-bit MBRs in preference to the 8-bit MBR. It is more likely that the two 4-bit MBRs can be placed with reduced competition for routing resources with the intervening register. Large MBRs may reduce the register count but can create routability problems when placed close to other MBRs, and their large area can increase the placement difficulty [26]. In future work, we plan to explore the possibility that, instead of penalizing the MBR candidates with obstacle registers through an increased weight, we completely remove them from the candidate list.

Fig. 7 summarizes the weights for the MBR candidates of the compatibility graph of Fig. 5, with placement per Fig. 6. When no incomplete MBRs are allowed, cliques {B, F} and {A, C, D} are mapped to 3-bit MBRs, while cell E is kept separate. This solution reduces the initial six registers to three. When incomplete MBRs are allowed, the same final register count is achieved with a different final outcome. Both choices in Fig. 7 minimize the ILP cost function, and allow the three final MBRs to be placed in distinct regions without intersecting each other. Before placement legalization, the composed MBRs may overlap with other non-register cells in the region, but the weights assigned to each candidate reduce the chance of overlapping with neighbouring MBRs. This example highlights the option of incomplete MBRs. In practice, the incomplete register AE will be rejected, as its area is significantly larger than the area of the registers it replaces.

Candidate MBRs involve both the initial registers of the design and the ones derived after MBR decomposition. As described in Section III, the MBRs resulting after decomposition are placed at the position of the decomposed MBR. This initial placement decision does not limit the creation of large MBRs, as verified experimentally, since the test polygon for every candidate clique/MBR covers only compatible registers. The decomposed MBRs lead to some compatible registers and some incompatible ones (this is the reason why we decomposed in the first place). So the incompatible ones do not block the formation of larger cliques.

### *E. MBR Mapping*

The ILP selects candidate MBRs that minimize the total number of registers and are less intertwined in the layout. For each MBR, the ILP selects just its bit width and the functional class of cells to which it belongs. Two further steps are needed for MBR assignment: MBR mapping and placement.

We must map the assigned MBR to a specific library cell. From the functional compatibility checks performed earlier, we know there is a compatible MBR in the library. From the available MBRs, we should select the one that best fits the timing and the drive strength profiles of the registers that it replaces.

The drive strength of the selected MBR should match the maximum drive strength of the registers that will be replaced by the MBR. This avoids degrading the timing of the design, but may incur an area and power overhead. However, since the registers to be merged are already drive strength compatible, the area overhead is avoided.

Registers with a high drive strength but a large timing slack (checked during drive-strength compatibility) don't determine the drive strength of the new MBR. In this manner, those registers are implicitly downsized and their slack is reduced.

Any extra area paid depends on the difference of the drive strength between the composed registers versus how many control pins are shared by the MBR. To minimize clock power, we select the MBR with the lowest pin capacitance from the MBR library cells that closely match the drive strength of the registers to be replaced by the MBR. Due to the large variety of MBR cells in modern libraries, if the drive strength and the clock pin capacitance are not appropriately selected, they may cancel the benefits of MBR composition by creating significant timing problems or diminishing the clock tree power reduction.

MBR mapping also ensures that the scan chain definitions encoded as scan compatibility constraints are preserved with the lowest possible cost. MBRs with multiple scan in/out pins may seem attractive as their area and power are lower than their counterparts with internal scan. In reality, MBRs with multiple scan in/out pins incur the extra routing resource cost of the external scan chain connectivity. For this reason, MBR library cells with external scan chains are avoided during MBR selection—they are typically selected only when there is no other alternative, or for mapping registers that are non-consecutive and belong to an ordered scan section.

### *F. MBR Connection and Placement*

After mapping to the assigned MBR, we must determine a location for the new cell, and connect the input and output nets of the MBR to the many available pins.

We first assign nets to pins of the MBR. We topologically sort the replaced cells by their horizontal position, and connect the pins of the leftmost cell to the leftmost bit of the MBR. This is done for the input-D and output-Q pins, as well as any control pins. If these are registers on an ordered scan chain, the order of pin assignment is dictated by the scan order.

The new MBR is placed in the position that minimizes the length of the wires connected to its D and Q pins. To identify the best location, we use a linear programming (LP) approach.

The new MBR must be placed in the common timing-feasible region of the compatible cells. For each D/Q pin of the replaced registers, we identify their fan-in and fan-out pins to which the MBR will connect, respecting the connectivity of the original registers.

For each MBR pin and its connections, we create a bounding box. We reference each MBR pin's coordinates relative to the MBR's lower left corner, plus some offset ($dx_i$, $dy_i$) for the pin's location on the cell. The lower left ($x$, $y$) coordinates of the MBR are the variables to be determined by the LP.

For the bounding box that corresponds to the input or output connections of each pin $i$, we use the half-perimeter wire-length to estimate the wire-length of the new wires. For each bounding box, the approximate wire-length $wl_i$ is

$$\begin{aligned} wl_i &= \max\{x_h, x + dx_i\} - \min\{x_l, x + dx_i\} \\ &\quad + \max\{y_h, y + dy_i\} - \min\{y_l, y + dy_i\} \end{aligned}$$

where ($x_l$, $x_h$, $y_l$, $y_h$) are the bounding box coordinates, and ($x$, $y$) are coordinates of the MBR's lower left corner. We use a LP to minimize the wire-length of the D/Q pins of the MBR:

**Minimize** $\sum_{i=1}^{M} wl_i$
**Subject to** ($x$, $y$) ∈ MBR's timing-feasible region

The max and min functions in the objective are removed by the use of extra helper variables. For example, max $\{x_h, x+dx_i\}$ is transformed to inequality constraints $x_h \le z$ and $x+dx_i \le z$, while the opposite inequality is used for the min function.

Every new MBR replaces the set of merged registers, and its placement reuses the space freed by them. The drive strength compatibility check ensures that the area of the replaced cells is enough to contain the area of the larger composed MBR. However, the reorganization of this freed space for placing the new MBR causes the rest of the pre-placed gates in the same region to move slightly. As we verified experimentally, the legalization step that follows MBR placement, along with the incremental timing-driven optimization performed by default after legalization, manages to handle the movement of the rest of the gates without any true disturbance to timing, while preserving the desired wire length reduction.

## V. Post MBR- Composition Steps

After MBR composition, there are two further optimizations. As shown in Fig. 2, these are MBR downsizing and the recovery of incomplete MBRs, with placement legalization in between. Downsizing reduces the clock pin capacitance and the area of the MBRs by sizing them down to the point that does not degrade timing. Recovery of incomplete MBRs better utilizes the unused pins of incomplete MBRs by locally merging compatible MBRs with incomplete MBRs.

**Algorithm 1: MBR downsizing**

```
foreach register ∈ MBRs do
    prevTNS ← TNS; prevWNS ← WNS;
    while register is downsizable do
        Downsize register;
        Run Incremental STA;
        if prevTNS ≤ TNS && prevWNS ≤ WNS then
            prevTNS ← TNS; prevWNS ← WNS
        else
            Undo downsize; Break;
        end if
    end while
end for
```

### *A. MBR Downsizing*

During MBR mapping, we selected for each new MBR the library cell that best matched the drive strength of the replaced registers, without performing any additional optimization that would trade off timing slack with MBR area and clock pin capacitance. However, based on the timing profile of each register, significant clock pin capacitance and area can be saved by downsizing the non-timing-critical MBR cells.

Downsizing the MBRs uses a brute-force approach, avoiding violating either the Total Negative Slack (TNS) or the Worst Negative Slack (WNS) of the design, shown in Algorithm 1.

For each MBR, we find the set of equivalent library cells. The cells are sorted in descending order based on their drive strength. We test the cells with lower drive strength than the examined MBR. The MBR can be downsized if there is at least one more cell with a size smaller than the examined MBR that has not yet been tested. After every downsizing, the TNS and the WNS generated by this change must not be worse than the TNS and WNS using the previous gate size. We stop searching when the MBR resize degrades TNS and WNS, in which case we keep the size from the previous round.

### B. *Recovery of the unused pins of incomplete MBRs*

After the placement has been legalized, we identify local compatible registers that can be merged with the incomplete MBRs to better utilize their disconnected pins. For example, if an 8-bit incomplete MBR with one empty bit is next to a compatible single-bit register, we can remove the single-bit register and connect its nets to the pins of the empty bit of the 8-bit MBR. This reduces both the number of incomplete MBRs and the total number of registers.

**Algorithm 2: Recovery of incomplete MBRs**

```
foreach register ∈ Incomplete MBRs do
    pins ← NumberOfUnusedPins (register);
    nearbyRegs ← Registers inside fixed size window
    foreach s ∈ nearbyRegs do
        if !isCompatible (s, register) or
            NumberOfPins(s) > pins then
            Remove s from nearbyRegs;
        end if
    end for
    Sort nearbyRegs based on number of pins
    foreach s ∈ nearbyRegs do
        if NumberOfPins(s) ≤ pins then
            Connect Nets(s) to the unused pins of register;
            pins ← pins – NumberOfPins(s);
            Remove s from the design;
            if pins == 0 then Break;
        end if
    end for
end for
```

The steps in the recovery of incomplete MBRs are shown in Algorithm 2. First, we find all the MBRs that have unconnected pins. For each incomplete MBR, we find all the compatible registers that are placed inside its timing-feasible region. We consider only registers with bit width less than the empty pins of the incomplete MBR. In this way, we don't need to break down existing MBRs to reuse their pins; we can just reconnect the nets of the smaller nearby registers.

All candidate registers are sorted in descending order based on their number of pins. If two registers have the same number of pins, the register that is closer to the MBR is ordered first. Next, while there are empty pins in the examined MBR, we determine if we can completely remove a nearby compatible register and connect its nets to the unconnected pins of the MBR. If all pins of the MBR are connected, we move to the next incomplete MBR. Note that incomplete MBRs may still remain after checking all the nearby registers.

## VI. Experimental Results

The MBR composition methodology has been tested on six industrial benchmarks that are rich in MBRs after logic synthesis. Designs D1-D5 correspond to implementations at 28nm, while D6 is implemented in a 16nm process technology. The industrial designs used represent real use cases that were actually taped out. In all cases, the designs were optimized both in terms of physical layout density and timing. On average, the designs achieve an 80% layout density; above that, the designs are unroutable.

Our methodology aims to reduce the register count and clock tree capacitance, with only marginal disturbance to timing, wire length, and routing congestion. Before presenting the overall results for all designs, we would like to focus on two distinct cases that highlight how the proposed MBR composition flow performs under different scenarios. Two clock nets were selected in a placed design and CTS was performed on them, with and without the application of the proposed MBR composition flow. MBR composition was applied only on the registers driven by those clock nets.

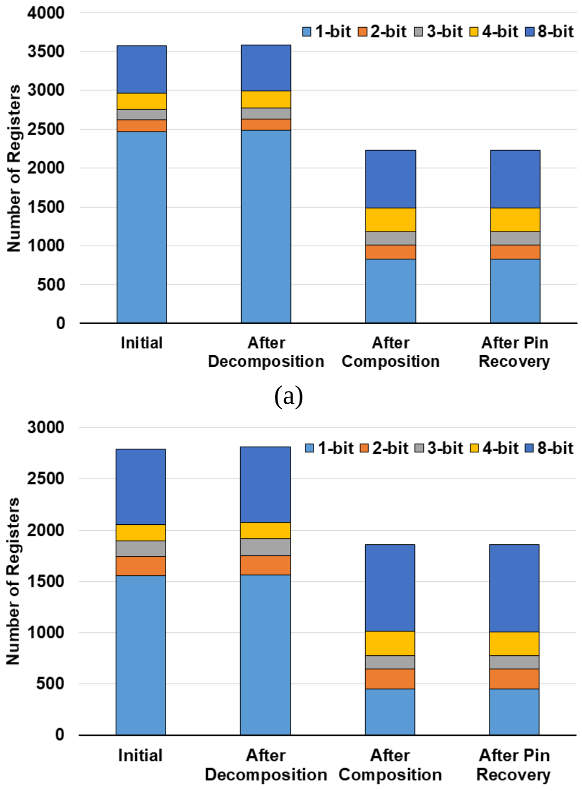
The first case (Case A) is a clock net that drives 3571 registers, which are mostly single-bit registers. This example shows the efficiency of the MBR composition step itself. Multiple compatible registers are composed to larger MBRs, allowing the use of incomplete MBRs.

The second case (Case B) is a clock net that has a similar fanout, consisting of 2795 registers, but a larger portion are existing MBRs formed in logic synthesis. This example highlights the power of MBR decomposition and optimization, combined with MBR re-composition, and sizing.

The initial configuration of Case A is shown in the initial bar of Fig. 8(a). The 3571 registers belong to several categories. The library used supports 1, 2, 3, 4 and 8-bit registers. After decomposition, the register count increases to 3588, due to decomposition of the MBRs with timing-incompatible pins. The MBR composition flow reduces the total number of registers by 37% relative to the number of registers in the initial design. Single-bit registers are reduced to almost a third of the initial count, and the number of 4 and 8-bit registers is increased accordingly.

Incomplete MBRs are allowed during MBR composition, provided that each incomplete MBR does not impose more than 5% area overhead relative to the area of the registers that it replaced. After MBR composition, there are 13 incomplete MBRs, all 8-bit MBRs. Following unused pin recovery, the final number of incomplete registers is reduced to 9, reducing the total number of registers to 2229.

Note that the efficiency of the proposed MBR composition flow is actually higher if we take into account that 1192 of the initial 3571 registers (almost a third) were skipped during the composition flow. They either do not match functionally to any MBR library cell, or there is no larger MBR in the library with



(a)

(b)

Fig. 8. The initial distribution of register sizes, and after each major step of the proposed flow for the clock net of (a) Case A and (b) Case B.

the same functionality. From the 3571 initial registers, only 2159 of them were composable, which was reduced to 1037 registers (2229 final registers, minus the 1192 skipped ones), corresponding to an almost 50% register reduction.

Similar results are derived for Case B, as shown in Fig. 8(b). With more MBRs initially, MBR decomposition touched more registers, and the 2795 initial registers increased to 2814 after decomposition. Despite this increase, MBR composition and unused pin recovery significantly reduced the total number of registers. After the application of the flow, the final number of registers is 1862 (with 21 incomplete MBRs), corresponding to 33% register savings. Again, not all registers were composable. Case B included 962 non-composable registers. Subtracting these from the register count, the number of registers saved by the proposed method was 50%.

The register reductions achieved directly translate to power, buffer, and wire length savings in the clock tree network. Table I summarizes the results achieved for Cases A and B after applying CTS on the initial clock nets and on those restructured by the proposed MBR composition flow.

Clock tree capacitance, wire length, and buffer count are all reduced in both cases examined. In Case A, the savings are higher due to the larger number of single-bit registers that allowed the MBR composition to explore more efficient MBR allocation choices. For example, clock tree capacitance and buffer count were reduced by 12% and 20%, respectively. In Case B, the composable MBRs were fewer and already had a larger bit width, limiting the available improvements for the proposed method. However, in addition to the 7% reduction in buffer count, measurable savings are still observed in clock tree capacitance and wire length (11% and 7.8%, respectively). The clock skew is also improved, by reducing the number of levels in the clock tree from 10 to 9 in Case A, and by reducing the clock wire length in both cases.

TABLE I
THE PROPERTIES OF TWO EXAMPLE CLOCK NETS OF THE INITIAL DESIGN AND THEIR PROPERTIES AFTER THE APPLICATION OF THE PROPOSED MBR COMPOSITION.

| Properties | Case A | | Case B | |
|---|---|---|---|---|
| | Initial | Proposed | Initial | Proposed |
| Max Leaf Levels | 10 | 9 | 9 | 9 |
| Rise/Fall Avg. Latency (ps) | 742 | 748 | 718 | 622 |
| Rise/Fall Avg. Clock Skew (ps) | 106 | 103 | 100 | 102 |
| Clock Wirelength (mm) | 32.0 | 31.1 | 34.2 | 31.5 |
| Buffer Count | 287 | 229 | 242 | 224 |
| Total Capacitance (pF) | 8.9 | 7.8 | 9.1 | 8.1 |

The clock leaf number and the clock pin capacitance reduction typically translate to lower clock latency. However, there are cases where the clock latency does not exhibit the expected reduction, due to the MBR placement algorithm that minimizes the total, and not specifically the clock tree wire length. As a result, MBRs might move away from their clock driver. In the case where the clock leaf defining the clock latency moves even further, a slight latency increase is observed. The same applies to clock skew, which is typically reduced; there are cases where slight increases can occur, due to the non CTS-friendly placement.

The MBR composition flow shown in Fig. 2 achieves similar improvements when applied across all the clock nets of the industrial designs. Also, we did not apply any additional useful skew offsets in our flow for these results.

Table II summarizes the register counts of the initial and the MBR-restructured designs. Even in designs with a large initial portion of MBRs, many registers were replaced by larger MBRs. The average savings in the total number of registers achieved by the proposed method is almost 30%. The largest savings are observed in design D2, which had the largest initial number of single-bit registers, while the smallest savings are observed in design D4, where the majority of the registers were already large 8-bit MBRs. The D6 number of single-bit registers actually increased, due to the initial MBR decomposition, but this permits more register composition with compatible slacks, resulting in an overall register count reduction of 20%.

The reported savings correspond to a significant improvement after taking into account how constrained the application of MBR composition is in real industrial designs that are rich in MBRs after logic synthesis. For example, in all examined cases, the compatibility graphs were extremely fragmented, due to the strict compatibility constraints. Each compatibility graph contained multiple independent subgraphs with sizes fewer than 10 nodes (recall that a node can correspond to an already

TABLE II
THE INITIAL DISTRIBUTION OF REGISTER SIZES AND AFTER THE PROPOSED FLOW.

| Initial | D1 | D2 | D3 | D4 | D5 | D6 |
|---|---|---|---|---|---|---|
| 1-bit | 21070 | 33918 | 19561 | 15478 | 19561 | 34938 |
| 2-bit | 1116 | 0 | 1453 | 3513 | 1453 | 4771 |
| 3-bit | 995 | 0 | 1726 | 1619 | 1726 | 2463 |
| 4-bit | 1196 | 3483 | 2409 | 4180 | 2409 | 48441 |
| 8-bit | 5445 | 0 | 9370 | 25148 | 9370 | 386 |
| #Registers | 29822 | 37401 | 34519 | 49938 | 34519 | 90999 |

| Proposed | D1 | D2 | D3 | D4 | D5 | D6 |
|---|---|---|---|---|---|---|
| 1-bit | 7739 | 12187 | 5202 | 9622 | 6446 | 36105 |
| 2-bit | 1576 | 1297 | 1340 | 2928 | 2445 | 3984 |
| 3-bit | 1226 | 1000 | 1393 | 1967 | 2213 | 3225 |
| 4-bit | 1842 | 2844 | 2220 | 4781 | 3785 | 9393 |
| 8-bit | 6402 | 1892 | 10761 | 25466 | 9390 | 19608 |
| Incomplete | 247 | 562 | 806 | 165 | 619 | 78 |
| #Registers | 19032 | 19782 | 21722 | 44929 | 24898 | 72393 |
| Savings | 36% | 47% | 37% | 10% | 28% | 20% |

formed MBFF during logic synthesis). These small subgraphs were more than 85% of the total subgraphs and covered around 60% of the total register bits of the designs. MBR candidate enumeration and the ILP-based MBR assignment combine to get the most out of these small subgraphs. In larger graphs (less than 5% of the subgraphs consist of more than 30 nodes), the proposed approach achieves equally good results, but with increased runtime.

Several other design characteristics of the initial and restructured designs are shown in Table III. By reducing the number of registers, MBR composition also reduces the complexity of the clock tree. Clock tree capacitance is reduced by 9.22% and buffer count is reduced by 11.45%, resulting in a similar reduction in clock power.

Clock and total wire length of the design is also reduced, due both to fewer registers and the wire-length-minimization-driven MBR placement. Note that in designs rich in MBRs, the clock wire length is a smaller percentage of the total wire length. The reduction of registers also led to a 0.37% and 1.64% average reduction in the total area of the designs and the total number of cells, respectively.

Although we perform significant circuit restructuring, on average we don’t increase the timing violations, as highlighted by the worst negative slack (WNS) and the total negative slack (TNS) of the presented benchmarks. On average, we reduce both WNS and TNS by 6.64% and 16.17% respectively. The

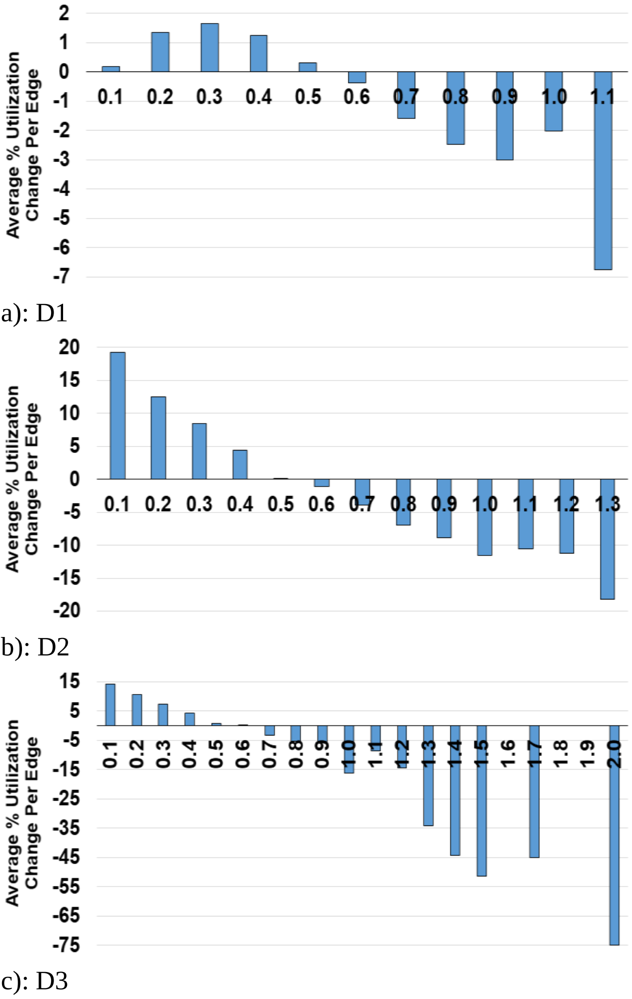
TABLE III
INDUSTRIAL DESIGNS CHARACTERISTICS BEFORE (INIT) AND AFTER THE PROPOSED MBR COMPOSITION FLOW (NEW).

| Design | | Total Area ($\mu m^2$) | Total #Cells | Total Wire-length (m) | WNS (ps) | TNS (ns) | Clock skew (ps) | Clock Wire-length (mm) | #Clock Buffers | Clock Cap (pF) | Over flow Edges |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D1 | Init | 598015 | 498434 | 14.19 | 47 | 1.82 | 176 | 669 | 2218 | 91 | 3 |
| | New | 596903 | 487646 | 14.01 | 42 | 0.48 | 194 | 613 | 1708 | 81 | 0 |
| | Save | 0.19% | 2.16% | 1.27% | 10.64% | 73.46% | -10.51% | 8.37% | 22.99% | 10.99% | 100% |
| D2 | Init | 952021 | 853912 | 16.67 | 719 | 865.33 | 196 | 453 | 3296 | 106 | 4 |
| | New | 943771 | 836690 | 16.57 | 616 | 869.18 | 181 | 416 | 2911 | 90 | 1 |
| | Save | 0.87% | 2.02% | 0.60% | 14.33% | -0.45% | 7.63% | 8.17% | 11.68% | 15.09% | 75% |
| D3 | Init | 1118762 | 666540 | 23.12 | 585 | 44.73 | 221 | 631 | 2825 | 132 | 382 |
| | New | 1118056 | 655151 | 23.76 | 665 | 69.75 | 189 | 603 | 2338 | 120 | 192 |
| | Save | 0.06% | 1.71% | -2.77% | -13.87% | -55.93% | 14.45% | 4.44% | 17.24% | 9.09% | 49.74% |
| D4 | Init | 2801693 | 1995640 | 65.04 | 2562 | 9018.63 | 264 | 1230 | 6459 | 272 | 46 |
| | New | 2800580 | 1984050 | 63.59 | 2548 | 9490.52 | 282 | 1219 | 6332 | 272 | 54 |
| | Save | 0.04% | 0.58% | 2.23% | 0.55% | -5.23% | -6.82% | 0.89% | 1.97% | 0.00% | -21.74% |
| D5 | Init | 1158447 | 693565 | 24.79 | 505 | 85.75 | 335 | 436 | 1875 | 92 | 354 |
| | New | 1152705 | 683582 | 24.80 | 435 | 42.20 | 233 | 417 | 1671 | 83 | 199 |
| | Save | 0.50% | 1.44% | -0.04% | 13.86% | 50.79% | 30.30% | 4.36% | 10.88% | 9.78% | 43.79% |
| D6 | Init | 653013 | 1188878 | 28.43 | 127 | 118.12 | 126 | 478 | 2762 | 192 | 0 |
| | New | 649263 | 1165887 | 28.49 | 109 | 77.51 | 126 | 437 | 2654 | 172 | 0 |
| | Save | 0.57% | 1.93% | -0.21% | 14.17% | 34.38% | 0.00% | 8.58% | 3.91% | 10.42% | - |

The final version of record is available at http://dx.doi.org/10.1109/TCAD.2018.2852740

only large discrepancy appears in design D3, where MBR composition leads to an increase in timing violations.

Design D3 includes several highly dense regions that also include timing-critical (start) endpoints that span across many paths of the design. When composing new MBRs in such dense regions, the rest of the cells in the region are slightly moved, even if they don't participate in the newly-formed MBRs. How far the rest of the cells move from their original positions depends on the internals of the detailed placement engine, and how it prioritizes each cell's type (sequential or combinational), its area, or its timing criticality. In the case of D3, the slight displacement of timing-critical cells that affect many timing endpoints (mostly single-bit register cells that didn't participate in MBR composition) causes a cumulative effect that increases the TNS of the design.



(a): D1

(b): D2

(c): D3

Fig. 9. The average percentage of change in the utilization per edge for each group of edges that initially exhibited similar utilization (globally). After MBR composition, initial low-utilization edges receive more load, thus increasing on average their utilization, while edges with high utilization on average receive less load.

There is no timing degradation if we avoid MBR composition in regions with excessive cell density. However, this approach leads to fewer composed MBRs, and to more registers in total at the end. The results given in the paper do not take such precautions, and every region (independent of its utilization) participates in MBR composition. This fully unconstrained approach decreases the total number of registers to 21722 (37% savings - initially there were 34519 registers) with a 50% TNS overhead relative to the initial design. When we skip the registers that belong to regions with more than 95% cell density, then the total number of registers becomes 22467 (34% savings relative to the initial design) while setup timing (WNS and TNS) is not degraded. Further decreasing the density cut-off threshold, as tested with additional experiments for 90% and 85% densities, increases register count without a noticeable difference in TNS.

A similar trend is observed regarding the final clock skew. The majority of the simplified clock trees produced after MBR composition demonstrate improved behavior with respect to clock skew, except in D1 and D4, where clock skew is marginally worse, due to an unfavorable placement of the MBR cells.

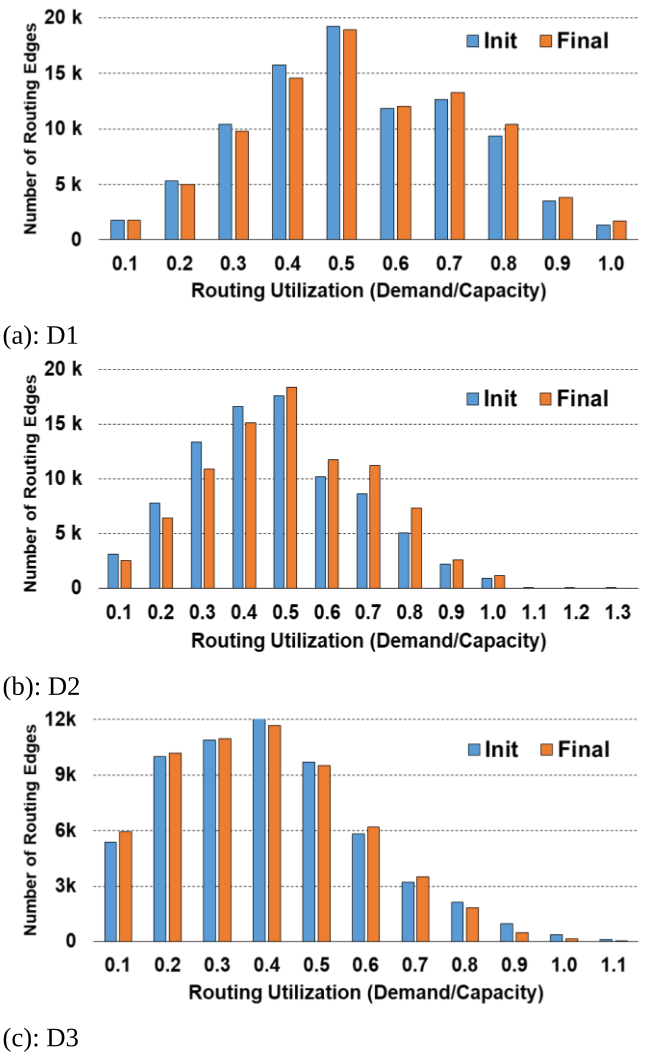
In this work, we do not aim at reducing the overall routing congestion of the design. Rather, our goal is to not degrade the global routing congestion profile of the initial design after applying MBR composition to reduce the number of registers and simplify CTS. This goal is achieved, as shown by the results reported in the last column of Table III. The difference in overflow edges [27], without and with our MBR composition methodology, is marginal due to the placement-aware weight selection for candidate MBRs in the ILP formulation.

In addition to the reduction of the number of overflow routing edges, as depicted in the rightmost column of Table III, the restructuring of the designs does not globally alter the routing congestion profile of their critical edges. The routing edges with high utilization (demand/capacity ratio) receive less utilization after MBR composition and detailed placement, while routing edges with low utilization are additionally loaded.

This behavior is highlighted in Fig 9(a) for all routing edges of D1. Each bar of Fig. 9(a) corresponds to the edges of the initial design that exhibit similar utilization, i.e., all edges that exhibit up to 0.1 utilization are covered in the leftmost bar of Fig. 9(a). For each of the initial utilization groups, Fig. 9(a) records the average percentage of change (positive or negative) to their utilization after MBR composition. Following the presented data, it is evident that the routing edges that initially exhibited low utilization have increased their utilization, and the ones with high utilization are less stressed after MBR composition. The same conclusion can be derived by the data presented in Figs. 9(b) and 9(c) for designs D2 and D3, respectively, while the rest of the designs exhibit similar behavior.

To isolate the routing congestion near the regions that include a MBR, we identified the routing edges that pass on top of or next to multi-bit registers in both the initial designs and the designs derived after applying the proposed MBR composition flow. For each edge, we recorded the routing utilization in the initial and final designs. The histogram of utilization for those routing edges is depicted in Fig. 10 for designs D1, D2 and D3. In all cases, the number of routing edges that are close to MBRs increases (due to the additional MBRs composed by the proposed flow), but their utilization remains under control, following the same trend depicted for all the routing edges of the design in Fig. 9.

Finally, Fig. 11 highlights the runtime of each part of the



(a): D1

(b): D2

(c): D3

Fig. 10. The histogram of the utilization of all routing edges that are close to MBRs in the initial designs and final designs after applying the proposed MBR composition flow.

MBR composition flow shown in Fig. 2. The flow was executed on an Intel Haswell server operating at 2.7 GHz with 512 GB of RAM. In all cases, MBR composition corresponds roughly to half of the runtime of clock tree synthesis (CTS), with the most expensive parts being the mapping of the selected MBRs and their legalization. The solution of the ILP, although consuming a non-negligible part of the overall runtime, is not the bottleneck for the application of the proposed MBR composition. On the contrary, the proposed enumerative ILP-driven methodology appropriately handles both the many small compatibility subgraphs and the small number of larger subgraphs, while significantly reducing the total number of registers.

## VII. Conclusions

Applying MBR composition on industrial benchmarks requires a balanced restructuring approach. In addition to the reduction in the number of registers and clock tree capacitance, it should also keep the potential degradation in slack, wire length, and routing congestion under control.

In this paper, we present a complete MBR composition flow that explores almost every aspect involved in the use of MBRs during physical design. MBR decomposition is introduced to partially alleviate the timing incompatibilities derived after the placement of the original netlist. Registers are then merged to form larger MBRs employing an ILP-based optimization that uses new and realistic rules that determine register compatibility. The ILP has a weighted selection of the best MBR candidates to facilitate their legalization and reduce contention for local routing resources. We permit incomplete MBRs to achieve additional MBR composition, but ensure that this is not detrimental for area or leakage.

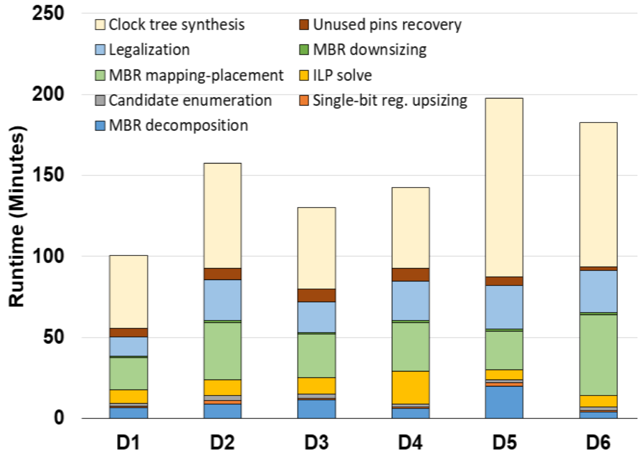


Fig. 11. The runtime of all the intermediate steps involved in the proposed MBR composition methodology including also the runtime of CTS.

The combined effect of these steps significantly reduces register count and, together with the timing-driven sizing of the MBRs, effectively reduces clock pin capacitance. The benefits are shown across six industrial benchmarks, demonstrating the effectiveness in producing a lighter clock tree without degrading timing or increasing routing congestion.

## Acknowledgments

We would like to thank Jared Anderson and Javier de San Pedro (Mentor, FR) for their valuable comments. I. Seitanidis is supported by the PhD scholarship of Alexander S. Onassis foundation.

## References

[1] D. Papa, N. Viswanathan, C. Sze, Z. Li, G.-J. Nam, C. Alpert and I. L. Markov, “Physical Synthesis with Clock-Network Optimization for Large Systems on Chips”, IEEE Micro 31(4) (2011), pp. 51-62.

[2] Monica Donno, Enrico Macii, and Luca Mazzoni, “Power-aware clock tree planning,” in Proc. of the 2004 International Symposium on Physical Design (ISPD), 2004, pp. 138-147.

[3] S. Roy, P. M. Mattheakis, L. Masse-Navette and D. Z. Pan, "Evolving challenges and techniques for nanometer SoC clock network synthesis," IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), 2014.

[4] Y. Cheon, P-H. Ho, A. B. Kahng, S. Reda, Q. Wang, “Power- aware placement,” in Proc. of the Design Automation Conference (DAC), Jun. 2005, pp.795-800.

[5] Y.-T. Chang, C.-C. Hsu, M. P.-H. Lin, Y.-W. Tsai, and S.-F. Chen, “Post-placement power optimization with multi-bit flip-flops,” in Proc. of the International Conference on Computer-Aided Design (ICCAD), 2010, pp. 218-223.

[6] W. Hou, D. Liu and P. H. Ho, “Automatic Register Banking for Low-Power Clock Trees”, in Proc. of the International Symposium on Quality of Electronic Design (ISQED), 2009, pp. 647-652.

[7] Y. Kretchmer, “Using Multi-Bit Register Inference to Save Area and Power: The Good, The Bad, and The Ugly”, EE Times Asia, 2001.

[8] D. Yi and T. Kim, “Allocation of multi-bit flip- ops in logic synthesis for power optimization,” in Proc. of the International Conference on Computer-Aided Design (ICCAD), 2016.

[9] A. B. Kahng, J. Li, and L. Wang, “Improved flop tray-based design implementation for power reduction,” in Proc. of the International Conference on Computer-Aided Design (ICCAD), 2016.

[10] I. H.-R. Jiang, C.-L. Chang, and Y.-M. Yang, “INTEGRA: fast multibit flip-flop clustering for clock power saving,” IEEE Trans. on CAD, vol. 31, no. 2, Feb. 2012.

[11] S.-H. Wang, Y.-Y. Liang, T.-Y. Kuo, and W.-K. Mak, “Power-driven flip-flop merging and relocation,” IEEE Transactions on CAD, vol. 31, no. 2, Feb. 2012.

[12] Y.-T. Shyu, J.-M. Lin, C.-P. Huang, C.-W. Lin, Y.-Z. Lin, and S.-J. Chang, “Effective and efficient approach for power reduction by using multi-bit flip-flops,” IEEE Transactions on VLSI Systems, vol. 21, no. 4, Apr. 2013.

[13] S. 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. Design Automation Electron. Systems, vol. 18, no. 3, Jul. 2013.

[14] C.-C. Tsai, Y. Shi, G. Luo, and I. H.-R. Jiang, “FF-bond: Multi-bit flip-flop bonding at placement,” in Proc. of the International Symposium on Physical Design (ISPD), 2013, pp. 147-153.

[15] ] M. P. H. Lin, C. C. Hsu, and Y. C. Chen, “Clock-tree aware multibit flip-flop generation during placement for power optimization,” IEEE Transactions on CAD, vol. 34, no. 2, pp. 280-292, Feb. 2015.

[16] S-C Lo, C-C Hsu, M. P-H Lin, “Power optimization for clock network with clock gate cloning and flip-flop merging,” in Proc. of the International Symposium on Physical Design (ISPD), 2014, pp. 77-84.

[17] S. Wimer, I. Koren, “Design Flow for Flip-Flop Grouping in Data-Driven Clock Gating,” in IEEE Trans. On VLSI Systems, vol. 22, no. 4, April 2014, pp. 771-778.

[18] C.-C. Hsu, Y.-T. Chang and M. P.-H. Lin, “Crosstalk-Aware Power Optimization with Multi-Bit Flip-Flops”, in Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC), 2012, pp. 431-436.

[19] T. Lee, D. Z. Pan, J-S Yang, “Clock Network Optimization with Multi-bit Flip-flop Generation Considering Multi-corner Multi-mode Timing Constraints,”, in IEEE Trans. on CAD, 2017, (to appear).

[20] H. Moon and T. Kim, "Design and allocation of loosely coupled multi-bit flip-flops for power reduction in post-placement optimization," in Proc. of the Asia and South Pacific Design Automation Conference (ASP-DAC), 2016, pp. 268-273.

[21] R. S. Shelar, “An efficient clustering algorithm for low power clock tree synthesis,” in Proc. of the International Symposium on Physical Design (ISPD), Mar. 2007, pp.181-188.

[22] G. Wu, Y. Xu, D. Wu, M. Ragupathy, Y.-y. Mo, and C. Chu, “Flip-flop clustering by weighted k-means algorithm,” in Proc. of Design Automation Conference (DAC), 2016.

[23] J. P. Fishburn, “Clock skew optimization,” IEEE Transactions on Computers, vol. 39, no. 7, pp. 945-951, Jul. 1990.

[24] Z.-W. Chen and J.-T. Yan, “Routability-constrained multi-bit flip-flop construction for clock power reduction,” Integration VLSI J., vol. 46, no. 3, Jun. 2013.

[25] C. Bron and J. Kerbosch, “Algorithm 457: Finding all cliques of an undirected graph,” Communications of the ACM, vol. 16, no. 9, Sep. 1973.

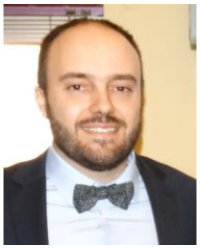
[26] W-K Chow, C-W Pui, E. F. Y. Young, “Legalization algorithm for multiple-row height standard cell design,” in Proc. of the Design Automation Conference (DAC), 2016.

[27] S. S. Sapatnekar, P. Saxena, and R. S. Shelar, Routing Congestion in VLSI Circuits: Estimation and Optimization. Springer, 2007.

**Ioannis Seitanidis** received the Diploma degree in electrical and computer engineering from the Democritus University of Thrace, Xanthi, Greece, in 2013, where he is currently pursuing the Ph.D. degree in computer engineering. He joined Mentor Graphics, France, in February 2018. His research interests include electronic design automation (EDA) algorithms, on-chip interconnection networks and computer architecture.

**Giorgos Dimitrakopoulos** received the B.S, MSc and Ph.D. degrees in Computer Engineering from University of Patras, Patras, Greece, in 2001, 2003 and 2007, respectively.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Democritus University of Thrace, Xanthi, Greece. He is interested in the design of digital integrated circuits, electronic design automation, and computer architecture, with emphasis in low-power systems design.



**Pavlos M. Mattheakis** received the M.S. and Ph.D. degrees in computer science from the University of Crete, Heraklion, Greece, EU, in 2007 and 2013, respectively. During his M.S. he was also with the Institute of Computer Science, FORTH, Heraklion, Greece, EU and with ISD S.A. Athens, Greece, EU. From 2007 to 2010 he was with Nanochronous Logic Inc, San Jose, CA, USA. During his Ph.D. he was also with the FORTH and the Technical University of Crete, Chania, Greece, EU. He is currently an R&D Engineer at Mentor Graphics, Grenoble, France, EU. His research interests include EDA algorithms for all stages of digital implementation and computer architecture. Dr. Mattheakis received the best paper award at ISPD14. He holds two patents.

**Laurent Masse-Navette** received an M.S. degree (Diplôme d’Ingénieur) in Computer Science from Institut National Polytechnique de Grenoble (INPG - ENSIMAG) and an M.A.S. degree in microelectronics from Université Joseph Fourier, Grenoble. After studying at INPG-CSI under professor G. Saucier between 1991 and 1994, he held positions as CAD and Research Engineer in the IC design and EDA industries at companies such as ST-Micro, Synopsys Inc. and Pulsic Ltd., where he built up expertise in the physical design and IC layout implementation domains, with a special focus on Clock Tree Synthesis. He joined Mentor Graphics in 2010 where he led the Clock Tree Synthesis R&D group and is now focusing on an Architect role in the Nitro-SoC R&D group in Grenoble.

**David Chinnery** received a Ph.D. in Electrical Engineering from the University of California at Berkeley in 2006. He is the author of two books on Closing the Gap Between ASIC and Custom with tools and techniques for high-performance and low-power design. He is an author of two chapters in the EDA for Integrated Circuits Handbook, and various conference papers. David has worked for the past six years at Mentor Graphics, where he is the R&D manager for the Optimization group of the Nitro place-and-route tool. Previously, David worked for five years in the CAD group at Advanced Micro Devices supporting both custom and synthesized microprocessor designs.