Timing Driven Incremental Multi-Bit Register Composition Using a Placement-Aware ILP formulation

Ioannis Seitanidis,
Giorgos Dimitrakopoulos
Democritus University of Thrace
ECE Dept., Xanthi, Greece

Pavlos Mattheakis,
Laurent Masse-Navette
Mentor, A Siemens Business
Grenoble, France

David Chinnery
Mentor, A Siemens Business
Fremont, USA

ABSTRACT

To reduce clock power, we present a novel timing-driven incremental multi-bit register (MBR) composition methodology for designs that may be rich in MBRs after logic synthesis. It identifies nearby compatible registers that can be merged without degrading timing, and without reducing the "useful clock skew" potential. These registers are merged providing the MBR placement can be legalized according to the proposed simplified physical constraints. A new integer linear programming (ILP) formulation minimizes the total number of registers in the design. It significantly reduces register count and clock capacitance, without adding any timing/routing/placement violations and without increasing the total wire-length of the designs, as shown by experimental results on industrial benchmarks.

1 INTRODUCTION

Reduced power consumption is a key design criterion for modern circuit designs, 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 which 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 along with 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 $0.5fCV_{dd}^2$, for a capacitance C (dis)charging between 0V and supply voltage V_{dd} with switching frequency f. Clock gating reduces the switching frequency. Placing registers together in clusters reduces the wire capacitance load on the clock network [2, 3]. Merging registers to multi-bit registers (MBR) further reduces the switching capacitance.

Multi-bit register (MBR) composition reduces the complexity of the clock tree by reducing the number of clock sinks, which in turn minimizes the clock tree's wire length and as a result reduces 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 allows a smaller clock tree to be used, with fewer and smaller clock buffers, further reducing the clock power.

Compatible registers that can be either single-bit flip flops or latches, or MBRs composed during logic synthesis, are grouped to larger MBRs with the goal to reduce register count and effectively achieve a lower clock tree power. To be compatible, they must be clocked by the same clock signal, including any clock gating logic. They must also be of the same register functional type. If the scan connectivity is internal to the MBR, they must also be in the same scan partition to be on the same scan chain.

Moving registers from their original locations to the location of the new MBRs may degrade timing slack, wire length, or routing congestion. Therefore, MBR merging should carefully select which registers to merge to ensure that potential degradations in slack, wire-length, or congestion do not offset the power benefits of a lighter clock tree.

The proposed approach targets incremental 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 MBR composition methodology equally applies to circuits that initially have only single-bit registers or that are rich in MBRs identified earlier in the design flow [4]. The design's initial physical implementation provides a good estimate of the timing slacks to help determine the best groupings for MBR composition, while there is still room to apply further aggressive timing optimizations. By composing MBRs from registers with similar input/output slacks, we ensure that there is good opportunity to use "useful skew" clock arrival timing offsets to fix timing violations [5].

Related work focuses mostly on MBR composition using heuristic algorithms on designs that initially have only 1-bit or at most 2-bit registers. In most cases, MBR composition is applied deeper in the physical design flow, after detailed placement [6–11] or post-CTS [12]. A few approaches have introduced MBR composition earlier in the flow [4, 13]. The late application of MBR composition narrows the design space to identify candidate MBRs, and each new choice requires incremental legalization and rebuilding the clock tree if after CTS. This results in long runtime and can cause timing hotspots with the disturbance of the clock sink points. Too early application of MBR composition misses critical placement and timing information that affects the final result.

Therefore, we argue for an incremental MBR composition that can be applied in various phases of the design flow to increase the effectiveness of MBRs. In this work, we target industrial designs which already have many MBRs after logic synthesis and apply on them a novel MBR composition methodology with the following properties:

- MBR composition merges only registers that are compatible in terms of functionality, timing, and placement. Additionally, the registers replaced by an MBR exhibit similar

DAC'17, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/3061639.3062327

input/output slacks, thus enabling the application of the same useful clock skew after CTS.
- MBR allocation 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 placement legalization.
- Incomplete MBRs, where some D/Q pin pairs are left tied-off/disconnected, are allowed for the first time. This option tackles the granularity limitations regarding the bit-width of MBRs in typical standard cell libraries. It further reduces the register count, without negatively affecting the area or leakage power.

The rest of the paper is organized as follows. In Section 2, we will first discuss the compatibility criteria that determine which registers can be composed into MBRs. Section 3 details the methodology for enumerating MBR candidates and introduces the ILP formulation that minimizes the number of registers. Then in Section 4, we specify the placement and mapping of the assigned MBRs to specific cells of the library. Section 5 presents the experimental results, and conclusions are drawn in Section 6.

2 REGISTER COMPATIBILITY

Our first goal is to identify the registers in the design that could be replaced by an equivalent MBR. A single or multi-bit register can be replaced by a larger MBR only if there is one in the library with equivalent functionality. For example, a register with a reset pin can be replaced only if an 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.

Although a group of registers (or latches) have 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 if the registers are *compatible* in terms of functionality, scan chain organization, placement, and timing profile.

Registers are *functionally compatible* when their control pins, such as reset or scan-enable, are driven by the same nets of the circuit, and they share the same clock gating enable 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. 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.

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 either have 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 and moving scan pins across different scan chains is allowed, then no additional constraints are imposed because of the scan chain definitions. If there are scan chain order constraints, then new compatibility constraints are introduced. For registers that belong to ordered scan sections, they 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.

For cells in the same functional class, *placement compatibility* is checked similar to [9]. 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. Nearby registers are placement compatible if their regions overlap, providing a shared region where the MBR can be placed. This is checked on a placed design to give a rough but realistic sense of the relative placement and distance between the registers under consideration. If the timing slack is negative, the feasible region is limited to the intersection of the bounding boxes of the violating pins with 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 compatibility checking, since it still creates a timing feasible region that matches its footprint, where other registers with positive slack can move.

After placement compatibility, *timing compatibility* is checked so that if multiple registers are replaced by a functionally-equivalent MBR, the resulting MBR to have similar input D-pin slacks and similar output Q-pin slacks. We should not compose cells that have positive D/negative Q slack with cells that exhibit negative D/positive Q slack. This composition could create opposite forces during the useful clock skew assignment to the MBR. A register with negative D slack favors increasing the clock arrival time to the MBR, while another with negative Q slack will be satisfied when the clock arrival time is reduced.

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 also must avoid very different clock useful skew values as only one can be realized for a given MBR and the difference will degrade useful skew opportunities for other timing paths to/from the MBR.

3 MULTI-BIT REGISTER COMPOSITION

The functional, timing, physical and scan compatibility of the registers is represented by the compatibility graph G. The graph nodes are the registers, whether single bit or pre-existing MBRs, and the edges of G reflect the compatibility between cells. An example compatibility graph is shown in Fig. 1. A MBR can be formed only by grouping together compatible cells. Therefore, the cells 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 can determine the set of candidate MBRs that can be examined during MBR composition. To enumerate all maximal cliques of G we use the Bron-Kerbosch algorithm [14]. For each maximal clique we enumerate all the valid sub-cliques following the possible sizes of the MBR library cells using a dynamic programming approach. The runtime complexity of maximal clique identification is $O(3^{n/3})$, which 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 subgraphs using K-partitioning. The partitioning is driven by the position of the register clock pins to maximize the clock tree power reduction obtained by MBR composition. Each subgraph

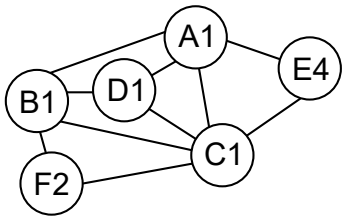

Figure 1: A compatibility graph of six registers. The compatible registers (nodes of the graph) are connected with an edge. Each register has a name and a size: A1 is a single-bit register, while E4 is a 4-bit MBR inserted during logic synthesis.

cannot exceed 30 nodes. Trying smaller bounds resulted in significant QoR (Quality of Results) loss in terms of composed registers, especially when the bound became smaller than 20 nodes. Increasing the bound further did not help either, as the additional runtime could not be justified by the slight QoR gains.

During clique enumeration, a clique is considered valid as long as the number of register bits involved 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 $\{A, C, E\}$ would be valid if we allow it to map to an 8-bit MBR. The 8-bit MBR would be incomplete since only 6 out of the 8 D/Q bits would be connected.

Incomplete MBRs may seem a waste of area and leakage since the circuit contains unused parts. In practice, it can still be advantageous, since MBRs allow for significant sharing of the control pins of the otherwise independent registers. For example, grouping 7 single bit registers, with common reset and clock signals, to an 8-bit MBR that uses one shared reset and clock wire saves at least 12 wire segments, even if one D/Q pin pair out of 8 are 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 properly connected to a scan chain.

Allowing incomplete MBR cells gives additional freedom to the MBR composition to minimize the total number of registers. However, to keep the area and leakage overhead under control, we consider an incomplete MBR as a valid candidate for MBR composition, only when the area per bit of the incomplete MBR is lower than the average area per bit of the underlying registers.

3.1 ILP Formulation

After the enumeration of all valid MBR candidates that are derived from the cliques of G, we need to assign the compatible registers to as few MBRs as possible. This is done using an integer linear programming formulation.

The candidate MBRs define a set $M = \{M_0, M_1, ..., M_k\}$ where each element M_i is a valid MBR candidate. Each compatible register participates 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 M_i, otherwise $a_{ij} = 0$.

The ILP should identify which candidate MBRs M_i will be selected at the end to replace the compatible registers of the design. To distinguish which M_i is actually selected from the other candidates, we introduce an additional 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. Thus, the total number of registers is minimized by solving the following integer linear program:

$$\textbf{minimize} \quad \sum_{i=1}^{|M|} w_i x_i$$

$$\textbf{subject to } \forall \text{ register } j : \sum_{i=1}^{|M|} a_{ij} x_i = 1, \qquad a_{ij}, x_i \in \{0, 1\}$$

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.

3.2 Weights to limit wire-length & congestion

By weighting MBR candidates, we limit the increase in routing congestion and wire-length during MBR composition. Without this, both routing congestion and wire-length can significantly increase. Experimental results in Section 5 show that this weighting heuristic keeps them both under control.

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 favorable MBR candidate, with the smallest weight, is the one that avoids any other closely-placed compatible registers that do not participate to the examined clique and could belong in another composed MBR. In this way, we limit the probability that the nets of the two new MBRs cross each other, thus keeping routing utilization under control.

For each MBR candidate (a clique in the compatibility graph), we define a polygon by the corners of the participating registers. The polygon defined by the nodes of the clique under examination should define a contiguous area. We use the convex hull formed by the outer corner points of the registers under examination.

Fig. 2 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 favorable 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, so smaller movement of fewer registers helps minimize the placement disturbance.

For the composition of a 3-bit MBR from $\{A, B, C\}$, we observe in Fig. 2 that the polygon defined by the corners of A, B, and C includes register D. The composition of this 3-bit MBR is less favorable since register D may end up merging with another MBR that will be closely placed with the 3-bit MBR and increase locally the utilization of the routing resources.

Therefore, by weighting appropriately 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. In contrast, when there are many intervening registers, we promote the selection of smaller but clean MBRs. This two-faceted

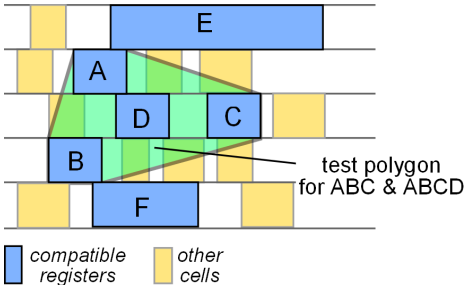

Figure 2: This shows the initial placement of registers in the Fig. 1 compatibility graph. The size of each register corresponds roughly to its bit width (number of D and number of Q pins). To improve routability after mapping to MBRs, we check inside the surrounding polygon of the clique for the presence of other register. The fewer intervening registers, the more favorable the candidate MBR.

goal is achieved with a heuristic weight w_i for each candidate MBR M_i as follows:

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

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 registers that block the convex polygon defined by the outermost corners of the registers replaced by M_i. A register counts as a blocking register for M_i if its center is inside the corresponding test polygon and it is not a constituent register of M_i. For the example shown in Fig. 1, clique $\{A, B, D\}$ has $\{b_i, n_i\} = \{3, 0\} \Rightarrow w_i = 1/3$ since it is not blocked by any other register in Fig. 2, whereas, the clique $\{A, B, C\}$ has $\{b_i, n_i\} = \{3, 1\} \Rightarrow w_i = 6$ as the center 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 registers, the weight promotes the selection of larger MBRs. For instance, the weight of a clean 8-bit MBR is $\frac{1}{8}$, which is smaller than the weight of two clean 4-bit MBRs, i.e., $\frac{1}{4} + \frac{1}{4}$, 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 more smaller MBRs. Large MBRs reduce the register count but can create routability problems when placed close to other MBRs and their large area can significantly increase the placement difficulty. Assume for example the case of an 8-bit MBR candidate that has one obstacle register, i.e., $\{b_i, n_i\} = \{8, 1\}$. In this case, the weight of this candidate would be $w_i = 16$. The equivalent choice with two smaller 4-bit MBRs would be to 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 would be equal to 8.25 that would make the ILP select the two 4-bit MBRs relative to the one 8-bit MBR. In this way, it is more likely that the two 4-bit MBRs can be placed without competing for routing resources with the intervening register.

The weights that correspond to all MBR candidates of the compatibility graph of Fig. 1 and their placement shown in Fig. 2 is summarized in Fig. 3. When no incomplete MBRs are allowed $\{B, F\}$ and $\{A, C, D\}$ are mapped to 3-bit MBRs, while cell E is kept separate. This reduces the initial six registers to three. When incomplete MBRs are allowed the same final register count is achieved

MBR candidates and their weights

Original		2-bit		3-bit		4-bit		5-bit		6-bit	
A	1.00	AB	0.50	BF	0.50	ABCD	0.25	AE	0.20	AEC	0.17
B	1.00	AD	0.50	CF	0.50			BCF	8.00		
C	1.00	AC	0.50	ABD	0.33						
D	1.00	BC	4.00	BCD	0.33						
E	1.00	BD	0.50	ABC	6.00						
F	1.00	CD	0.50	ADC	0.33						

Mapped to an 8-bit incomplete MBR

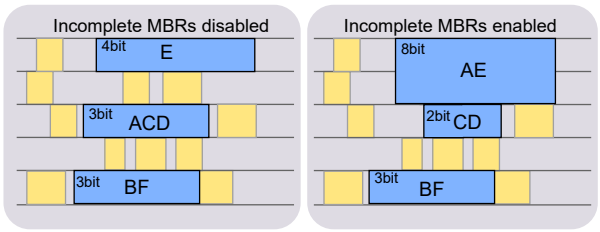

Figure 3: The weights of the candidate MBRs and the selected solution. The 5-bit and 6-bit MBRs can be mapped only to an 8-bit incomplete MBR.

with a different final outcome. Both choices minimize the cost function of the ILP, and allow the three final MBRs to be placed in distinct regions without intersecting with one another. Before placement legalization, the composed MBRs may overlap with other non-register cells in the region defined by the surrounding polygon of the corners of the participating compatible registers, but the weights assigned to each candidate reduce the chance of overlapping with neighboring MBRs. Please note that this example highlights on purpose the possibility of using incomplete MBRs. In reality, incomplete register AE would have been rejected since its area is significantly larger than the are of the registers it replaces.

4 MBR MAPPING AND PLACEMENT

The placement-aware ILP selects from the set of candidate MBRs those that minimize the total number of registers of the design and that are less intertwined in the layout. For each MBR, the ILP has selected just its bit width and the functional class of cells it belongs to. Two further steps are needed for a valid MBR assignment: MBR mapping and placement.

4.1 MBR Mapping

At this step, we need to map the assigned MBR to a specific MBR cell of the library. From the compatibility checks performed earlier, there is certainly a functionally compatible MBR in the library. From the available MBRs, we should select the one that fits best in the timing and the drive resistance profile of the registers that it replaces. With *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 standard cell library timing models.

The drive resistance of the selected MBR should match the minimum drive resistance of the registers that will be replaced by the MBR. This avoids degrading the timing of the design, albeit with some possible area overhead. The extra area paid depends on the difference of the drive resistance between the composed registers versus how many control pins are shared by the MBR. From the MBR library cells that match closely the drive resistance of the registers to be replaced by the MBR, we select the MBR with the lowest clock pin capacitance to minimize clock power.

Due to the large variety of MBR cells in modern libraries, if the drive resistance and the clock pin capacitance are not selected appropriately, they may cancel the benefits of MBR composition by creating significant timing problems or diminishing the clock tree power reduction.

Additionally, register mapping 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 typically smaller compared to their counterparts with internal scan chain. In reality, MBRs with multiple scan in/out pins have the extra cost of the external scan chain which consumes significant routing resources. For this reason, MBR library cells with external scan chains are penalized during MBR selection. They are typically selected only when there is no other alternative, to map registers which are non-consecutive but belong to an ordered scan section.

4.2 MBR Placement

After mapping to the assigned MBR, we need to identify the suggested location for the new cell. We target the position that minimizes the length of the connected wires on the D and the Q pins of the MBR, using a linear-programming approach.

The new MBR should be placed in the common timing feasible placement region of the compatible cells. For each D/Q pin of the compatible cells that will be replaced, we identify their fan-in and fan-out pins. Each MBR pin connects to some of those identified pins, respecting the connectivity of the original registers.

For each pin of the MBR and its connections, we create a bounding box assuming the coordinates of the pin of the MBR as unknown. For each pin we don't create a separate set of unknown variables, but we reference each pin's coordinates relative to the coordinates of the MBR's lower corner plus some offset in both dimensions (dx_i, dy_i) depending on the pin's location inside the cell. Thus, only the cell corner's coordinates need to be found.

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

$$wl_i = (\max(x_h, x + dx_i) - \min(x_l, x + dx_i)) + (\max(y_h, y + dy_i) - \min(y_l, y + dy_i)),$$

where (x_h, x_l, y_h, y_l) are the coordinates of the box boundaries for each pin, and (x, y) are the coordinates of the MBR's corner. We use a linear program to minimize the wire-length of the D/Q pins of the MBR as follows:

$$\textbf{minimize} \quad \sum_{i=1}^{|M|} wl_i$$

$$\textbf{subject to } (x, y) \in \text{MBR's timing feasible region}$$

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 two inequality constraints $x_h \leq z$ and $x + dx_i \leq z$, while the opposite inequality is used for the min function.

5 EXPERIMENTAL RESULTS

The MBR composition methodology has been tested on five 28nm industrial benchmarks that are rich in MBRs after logic synthesis. Our incremental methodology aims to reduce the register count and clock tree capacitance, with only marginal disturbance to timing, wire-length, and routing congestion. Table 1 shows the initial characteristics of the designs after placement and optimization in the rows labeled 'Base'. From the total number of registers (column 'Total-Regs') only a small subset can be composed to larger MBRs (column 'Comp-Regs'), because some registers (a) are specified as non-modifiable by the designer, (b) have no logically equivalent MBR in the library or (c) represent already the largest possible MBR in their functional-equivalence class.

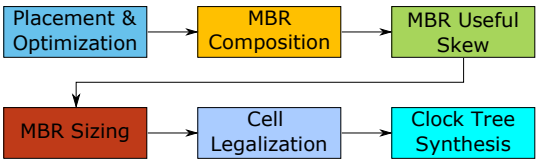

Figure 4: The implementation flow with MBR composition.

The implementation flow applied is shown in Fig. 4. After placement and optimization, MBR composition and optimization are performed, incrementally on MBRs already introduced after synthesis. Then, useful skew is applied to the new MBRs, benefiting from their timing compatible smaller counterparts. Useful skew improves the worst slack in each new MBR, broadening the solution space that can be examined by MBR sizing. As a result, both MBR area and clock pin capacitance are further reduced. Incomplete MBRs are also allowed, provided that each incomplete MBR, does not impose more than 5% area overhead relative to the area of the registers that it replaced. The obtained results are summarized in rows 'Ours' of Table 1.

The proposed methodology greatly reduces the total number of registers. Total register count drops by 29% on average (each MBR, independent of its size, counts as one register). Counting only the composable registers where MBR composition can be applied, the average reduction is 48%. Clock tree capacitance is reduced by 6% and buffer count is reduced by 4%, giving a similar reduction in clock power. The 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. With respect to runtime, the new proposed steps of MBR composition and optimization account on average for 60 min CPU time on Intel Haswell server @ 2.7GHz with 512 GB RAM.

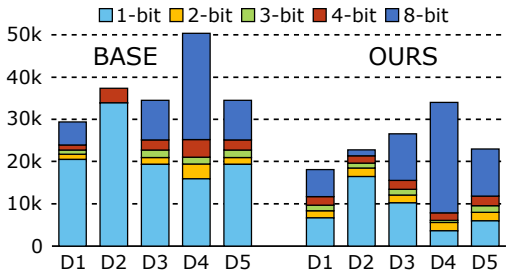

Figure 5: Bit widths of MBRs before & after MBR composition.

Although we perform significant circuit restructuring, we don't increase the timing violations, as highlighted by the TNS and the number of failing timing endpoints. Routing congestion is also not degraded. On average, failing endpoints are about 38% of the total endpoints for these designs. The difference in overflow edges [15], without and with our MBR composition methodology, is marginal

Table 1: Industrial design characteristics before and after MBR composition.

Design		Cells	Area (um^2)	Total Regs	Comp Regs	Wirelength (mm)		Clk Bufs	Total Clk Cap (pf)	TNS (ns)	Failing End-Points	Ovfl. Edges	MBR Exec. Time (min)
						Clk	Other						
D1	Base	485350	870621	29416	18332	231	14240	1202	66	41232	18285	7	
	Ours	470158	868102	18137	7050	216	14110	1088	60	41732	16138	2	45
	Save	3.1 %	0.3 %	38 %	61 %	6.0 %	1.0 %	9.5 %	9.0 %	-1.0 %	11.7 %	71 %	
D2	Base	853869	1228380	37401	27992	276	16430	1512	75	120895	53458	53	
	Ours	839457	1220290	22857	8853	261	16390	1218	64	93430	39959	52	68
	Save	1.7 %	0.7 %	39 %	68 %	5.0 %	0.2 %	19 %	15 %	22.7 %	25 %	2.0 %	
D3	Base	665369	1473670	34519	21880	395	23490	2138	90	70122	29350	519	
	Ours	652389	1470650	25453	7954	380	22840	1956	83	59653	28329	239	76
	Save	2.0 %	0.2 %	26 %	34 %	3.0 %	2.7 %	8.5 %	8.0 %	15 %	3.0 %	54 %	
D4	Base	1992432	3283820	50392	22017	1094	63680	6873	251	734379	271047	182	
	Ours	1980261	3284100	42535	16529	1083	63540	7040	252	684138	260053	164	40
	Save	0.6 %	0 %	15 %	25 %	1.0 %	0.2 %	-2.0 %	0 %	6.8 %	4.0 %	10 %	
D5	Base	695944	1473600	34519	21879	315	24850	1059	73	2990	50133	525	
	Ours	683008	1470780	22992	10076	295	24620	1014	66	2851	50087	467	80
	Save	2.0 %	0.2 %	33 %	54 %	6.0 %	1.0 %	4.0 %	10 %	4.6 %	0 %	11 %	

due to the placement-aware weight selection for candidate MBRs in the ILP formulation.

Fig. 5 shows the breakdown of the various MBR bit widths of all the registers of the design, before and after applying the proposed MBR composition methodology. More 8-bit MBRs are used up to a point where they don't create routing utilization problems. MBR composition in designs that already contain a large number of 8-bit MBRs, like D4, doesn't provide significant reduction in the clock tree capacitance. The clock tree capacitance in these cases is dominated by those 8-bit MBRs that were skipped during MBR composition and sizing. To optimize such designs, we plan in the future to consider the decomposition of the initial 8-bit MBRs and their recomposition using the proposed methodology, instead of skipping them completely, as done in this work.

For completeness, we compare the proposed ILP-based methodology with a heuristic-algorithm-based approach, similar to that performed in [8] and [12]. As shown in Fig. 6, the ILP gives better results for all designs, achieving 12% savings on average.

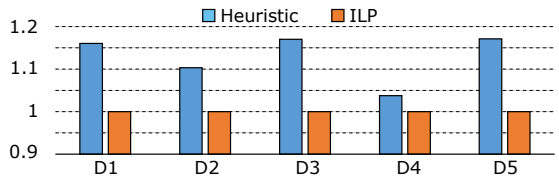

Figure 6: Normalized number of total registers when composition is performed using an ILP formulation and a maximal clique identification and MBR mapping heuristic.

6 CONCLUSIONS

Applying MBR composition on industrial benchmarks requires a balanced restructuring approach that, besides the reduction of the number of registers and clock tree capacitance, should keep the potential degradation in slack, wire-length and congestion under control. In this paper, we present for the first time a placement-aware ILP-based approach that satisfies the balanced restructuring approach, and can be applied incrementally both after global and detailed placement. The combined effect of the new rules that determine register compatibility, the weighted selection of the best MBR candidates, and the allowance of incomplete MBRs, gives significant reductions in register count and clock tree capacitance. This approach has been integrated into Mentor's Nitro place-and-route tool, achieving good results in an industrial design flow on modern designs.

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, C. Alpert, C. Sze, Z. Li, N. Viswanathan, G.-J. Nam, and I. Markov, "Physical synthesis with clock-network optimization for large systems on chips," *IEEE Micro*, vol. 31, no. 4, pp. 51–62, Jul. 2011.
[2] A. B. Kahng, J. Li, and L. Wang, "Improved flop tray-based design implementation for power reduction," in *Proc. of ICCAD*, 2016.
[3] 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*, 2016.
[4] D. Yi and T. Kim, "Allocation of multi-bit flip-flops in logic synthesis for power optimization," in *Proc. of the Int. Conf. on Computer-Aided Design*, 2016.
[5] J. P. Fishburn, "Clock skew optimization," *IEEE Transactions on Computers*, vol. 39, no. 7, pp. 945–951, Jul. 1990.
[6] 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.
[7] 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 ICCAD*, 2010.
[8] 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.
[9] 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.
[10] 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.
[11] 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. Des. Autom. Electron. Syst.*, vol. 18, no. 3, Jul. 2013.
[12] 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.
[13] C.-C. Tsai, Y. Shi, G. Luo, and I. H.-R. Jiang, "Ff-bond: Multi-bit flip-flop bonding at placement," in *Proc. of ISPD*, 2013, pp. 147–153.
[14] C. Bron and J. Kerbosch, "Algorithm 457: Finding all cliques of an undirected graph," *Commun. ACM*, vol. 16, no. 9, Sep. 1973.
[15] S. S. Sapatnekar, P. Saxena, and R. S. Shelar, *Routing Congestion in VLSI Circuits: Estimation and Optimization.* Springer, 2007.