Mixed Cell-Height Implementation for Improved Design Quality in Advanced Nodes

Sorin Dobre[†], Andrew B. Kahng[‡+] and Jiajia Li[‡]
UC San Diego, [‡]ECE and [+]CSE Depts., La Jolla, CA 92093, {abk, jil150}@ucsd.edu
[†]Qualcomm Technologies, Inc., San Diego, CA 92121, sdobre@qti.qualcomm.com

***Abstract*—In advanced nodes, standard-cell libraries can be developed with different cell heights (e.g., in FinFET technology, corresponding to different numbers of fins). Larger cell heights provide higher drive strengths, but at the cost of larger area and power consumption as well as pin capacitance. Cells with smaller heights are relatively smaller in area, but have weaker drive strengths and are more likely to suffer from routing congestion and pin accessibility issues. Existing design methodologies and tool flows are able to mix cells with different heights at the block level (i.e., each block contains cells of a particular cell height). To our knowledge, no design methodology in the literature mixes cells of different heights in a fine-grained manner. In this work, we propose a novel physical design optimization flow to implement design blocks with mixed cell heights in a fine-grained manner. Our optimization resolves the "chicken-and-egg" loop between floorplan site definition and the optimized choices of cell heights after placement. Comprehending the constraints and costs of mixing cells of different heights (e.g., the "breaker cell" area overheads of row alignment between sub-blocks of 8T and 12T cell rows), our optimization achieves 25% area reduction versus 12T-only implementation while maintaining the same performance, and 20% performance improvement versus 8T-only implementation while maintaining similar total cell area.**

I. Introduction

Standard cell-based implementation has been widely used for VLSI designs due to its relatively accurate abstraction and semi-regular layout. In advanced nodes, cells are designed with different heights (e.g., different numbers of fins in FinFET node). Larger cell heights have higher drive strengths at the cost of larger area and power consumption as well as pin capacitance. Smaller cell heights result in relatively smaller area, but have weaker drive strengths and are more likely to suffer from routing congestion and pin accessibility issues.[1] Figure 1 shows the delay and area tradeoff of buffers and inverters at foundry 28nm LP technology. In red are 12T cells, and in blue are 8T cells. We observe that 12T cells tend to achieve smaller delay at the cost of larger area.

Given that cells of different heights exhibit different tradeoffs among performance, power and area, mixing cells of different heights in a design is able to provide a larger solution space and improved design quality. Figure 2 shows the post-synthesis area and timing comparison among implementations of an open-source design AES [14] with 12T-only, 8T-only and mixed cell heights. Due to generally larger areas of cell instances (particularly those with low drive strengths), 12T-only implementation tends to have larger design area. On the other hand, weak drive strengths of 8T cells result in a large number of buffer insertions to meet timing constraints, which also increases design area.[2] We observe from the example that mixing cell heights achieves 14% and 18% area reduction at the post-synthesis stage versus the 12T-only and 8T-only implementations, respectively.

Motivated by the above observations, in this work we propose to mix cell heights at the sub-block level (i.e., within a single P&R block) of physical implementation, to achieve improved design quality – specifically, tradeoffs of achievable performance, power and area. However, optimizing a design by mixing cell heights is highly nontrivial. The challenges include the following.

[1]Although a cell with smaller height can be designed with large width to gain drive strength, the additional poly capacitance and metal capacitance can lead to area and power overheads as compared to a cell with the larger height.

[2]The larger total cell area of the 8T-only netlist as compared to that of the 12T-only netlist is due to tight timing constraints. We demonstrate in Section V that with loose timing constraints, 12T-only implementations typically incur area overhead as compared to 8T-only ones.

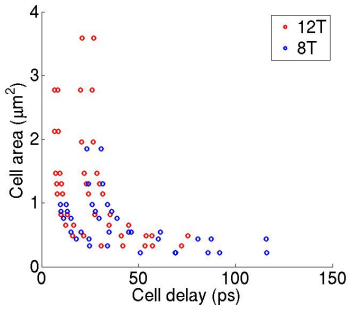


Fig. 1: Delay-area tradeoff of 8T and 12T buffers/inverters in 28nm LP foundry libraries. Load cap = FO4 + 20μm M3 wire.

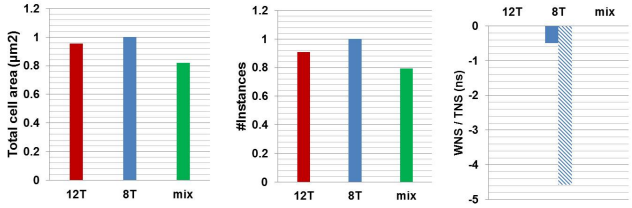


Fig. 2: Post-synthesis netlist with mixed cell heights has significant area reduction compared to 12T-only and 8T-only netlists. Technology: 28nm LP. Design: AES. Frequency: 1.5GHz. Corner: (SS, 0.95V, 125°C). Total cell area and number of instances are normalized to those of the 8T-only case. In the right figure, the solid bar indicates WNS, and the shaded bar indicates TNS. Implementations with 12T-only and mixed cell heights have no timing violations.

- Current design methodologies and tool flows can only mix cells of different heights at the block level, i.e., each block of a design uses cells with a particular height, with fine-grained mixing not available with today's EDA tools.
- There is a "chicken-and-egg" quandary: heights of cell rows are defined (in the placement site map) at the floorplan stage, but the optimized choices of cell heights are highly dependent on the placement outcome and timing constraints.
- There are costs associated with mixing cells of different heights. For instance, "breaker cells" must be inserted for row alignment between sub-blocks of cell rows with different heights.[3]

The contributions of this paper are as follows.

- To our knowledge, we are the first in the literature to propose mixed cell-height implementation at the sub-block or sub-island level in advanced nodes.
- We develop methodologies which can easily be integrated within existing physical design flow, using standard commercial tools, for mixed cell-height implementation.
- We show that mixing 12T and 8T cells in a 28nm LP foundry technology achieves 25% area reduction versus 12T-only implementation while maintaining the same performance, and

[3]We define a *breaker cell* as the space (i.e., placement and/or routing blockages) that must be inserted between the boundaries of regions of different cell heights.

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

20% performance improvement versus 8T-only implementation with similar total cell area.

II. Related Works

To our knowledge, there is no previous work on mixed cell-height design methodology within a block. However, the mixed cell-height placement problem bears some similarity to the problem of voltage island placement, in that both problems try to assign a certain attribute with different values (e.g., cell height or supply voltage) to standard-cell instances, in order to improve performance or reduce power. Further, both problems must comprehend a type of incurred cost (e.g., area overhead of "breaker cells" or insertion of level shifters) when performing the assignment. We therefore review exemplary works from the literature on voltage island placement.

Wu et al. [12] and Ching et al. [2] propose partitioning methodologies to divide post-placement die area into regions which will be assigned to different supply voltages. The objectives are respectively to minimize the number of partitions and to handle non-rectangular partitions. Wu et al. [13] further consider timing constraints during the voltage assignment. However, the interaction with standard-cell placement is missing in all of these works. An improved optimization proposed in [11] performs incremental placement to move timing-critical cells out from the low-voltage regions by adjusting net weights and cell delays. Guo et al. [5] embed their voltage-island-aware placement optimization to a partitioning-based placement algorithm to minimize the number of level shifters.

Although the voltage island placement problem has been well-studied in previous literature, there is still no available solution for mixed cell-height design implementation due to the existence of "chicken-and-egg" loop between floorplan and cell height selection, additional layout constraints, and area impact of cell height choices.

III. Problem Formulation

Table I gives notations used in the following discussion.

TABLE I: Notations used in our work.

Term	Meaning
h_i	available cell heights, $(0 \leq i \leq N$; h_0 is the minimum one)
(X^l, Y^b, X^r, Y^t)	coordinates of block area (i.e., standard-cell placement region)
P_j	partition in which cells are of the same height, $(0 \leq j \leq M)$
$(x_j^l, y_j^b, x_j^r, y_j^t)$	coordinates of partition P_j, $(0 \leq j \leq M)$
H_j	height of partition P_j, $(0 \leq j \leq M)$
W_j	width of partition P_j, $(0 \leq j \leq M)$
t_j	cell height corresponding to partition P_j, $(0 \leq j \leq M)$
w_{site}	placement site pitch (width)
d	shift of cell rows in vertical direction to avoid cell overlap

We state the **mixed cell-height placement problem** as follows. Given a design (i.e., gate-level netlist), timing constraints, Liberty and technology models, and floorplan constraints (i.e., P&R block area and aspect ratio bounds), place the design such that each cell instance is legally placed in a row with corresponding height. The objective of the placement is to achieve minimum design area while maintaining the (same) target performance.

Additional layout constraints for mixed cell-height implementation applied in our studies below are as follows.[4]

C_1: To ensure manufacturability, each region of a particular cell height must have at least two cell rows.

C_2: Due to N-well sharing, each region of a particular cell height must have an even number of rows.

C_3: Every region with a particular cell height must align with the block's overall metal and poly track definitions.

C_4: The horizontal distance between two regions of different cell heights must be no less than four placement sites.

C_5: The minimum vertical distance between two partitions of different cell heights must ensure that the power/ground (P/G) rail of one cell does not encroach beyond the P/G rail of another cell. (Figure 3 shows an example with M2 pitch = 64nm, and P/G rail width = 48nm and 64nm for 8T and 12T cells, respectively. Although the P/G rail width difference between 8T and 12T cells is less than one M2 pitch, to align cells to routing tracks, the minimum d in the example is 64nm.)

C_6: "Breaker cells" must be inserted to ensure the minimum horizontal and vertical distances between two regions of different cell heights.

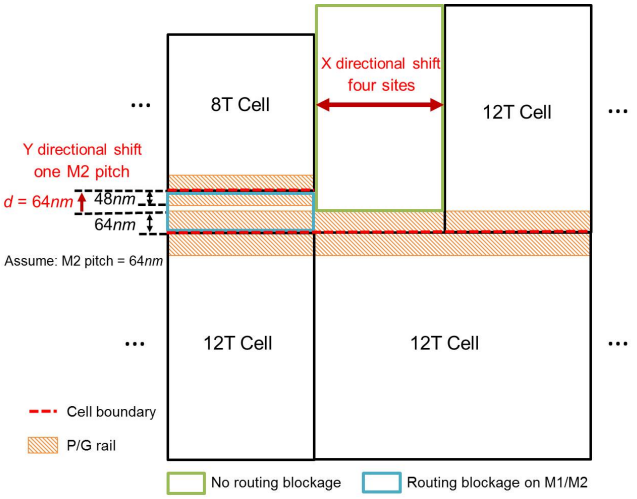


Fig. 3: Area cost of "breaker cells".

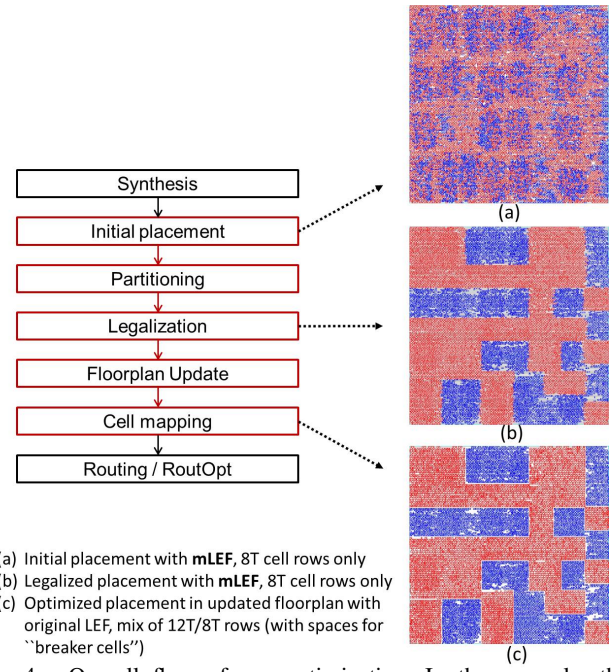
IV. Methodology

We now describe our optimization methodology for mixed cell-height implementation. The overall optimization flow is shown in Figure 4. Given an input design (i.e., RTL netlist) and timing constraints, we first synthesize it with Liberty files of all available cell heights having been made available to the logic synthesis tool. To resolve the "chicken-and-egg" loop between floorplan and cell height selection, we modify standard-cell LEF files such that all cells have the same height (i.e., the minimum cell height among all the available heights), while maintaining the original area of each cell.[5] In the discussion below, we refer to such modified LEF files as **mLEF**. In this way, we break the "chicken-and-egg" loop and enable a commercial placement tool to "freely" place cells with timing-awareness. Since we use the original Liberty timing/power models and preserve the original area for each standard cell (although with different aspect ratio of cell layout), this placement optimization is able to comprehend the tradeoff between timing constraints, power and area overheads. As a result, timing-critical cells tend to have larger heights (i.e., larger width with **mLEF**), while non-critical cells are smaller. An example initial placement solution of design AES is shown in Figure 4(a), in which 12T cells (with **mLEF**) are in red, and 8T cells are in blue.

Based on the initial placement solution, we partition the block area into regions of particular cell heights with awareness of area

[4]Our proposed approaches transparently handle other values of the parameters (e.g., minimum number of cell rows in a given-height region, or minimum separation between two different-height regions, etc.) mentioned here.

[5]In doing so, we round cell widths to the nearest whole site with no cell area reduction. E.g., given three libraries with heights 8T, 9T and 12T, (i) a 12T, 6-site cell would be represented by an 8T, 9-site cell; (ii) a 9T, 5-site cell would be represented by an 8T, 6-site cell; etc.



(a) Initial placement with **mLEF**, 8T cell rows only
(b) Legalized placement with **mLEF**, 8T cell rows only
(c) Optimized placement in updated floorplan with original LEF, mix of 12T/8T rows (with spaces for ``breaker cells")

Fig. 4: Overall flow of our optimization. In the example, the maximum cut number (K) = 30.

cost due to "breaker cells".[6] We then legalize the placement solution by (i) displacement of cells (i.e., placement perturbation) and (ii) swapping of cells across different heights (i.e., gate sizing). Here, we say that a placement solution is *legal* when each cell instance is placed in a region with the same height (e.g., as shown in Figure 4(b)). Once the placement solution is legal, we update the floorplan with space inserted to model the cost of breaker cells. In the end, we map cells to the cell rows of the updated floorplan and route the design (Figure 4(c)).

A. Floorplan Partitioning and Region Definition

We perform slicing-based partitioning using dynamic programming to divide the block area into regions of particular cell heights. Algorithm 1 shows our partitioning procedure. We first evaluate the cost of each candidate partition, i.e., $cost(x^l, y^b, x^r, y^t, 0)$, in which the fifth parameter indicates the number of cuts within the partition (Line 1). We define the height of a partition based on its majority cells, that is, the cell height with maximum corresponding total area of cells inside the partition is defined to be the height of the partition. We then estimate the cost of each partition as the sum of areas of minority cells in the partition (i.e., cells whose heights differ from the height of the partition). Figure 5 shows examples of partitioning solutions, in which the cost of a given 12T (resp. 8T) partition is the total area of 8T (resp. 12T) cells in the partition. Furthermore, we set the cost of a candidate partition to infinity if it violates any of the constraints (e.g., Constraints C_1 and C_2) described in Section III. More specifically, a partition (x^l, y^b, x^r, y^t) with height h_j must satisfy

$$y^t - y^b \geq 2 \cdot h_j \tag{1}$$

$$\lfloor \frac{y^t - y^b}{2 \cdot h_j} \rfloor \cdot 2 \cdot h_j \cdot (x^r - x^l) \geq (y^t - y^b) \cdot (x^r - x^l) \cdot U_j \tag{2}$$

Inequality (1) forces each partition to have at least two rows. Inequality (2) ensures that partitions in the updated floorplan, after rounding to an even number of rows per partition according to Constraint C_2, have enough sites to place cells; here, U_j is the placement utilization within the partition.

[6]Here cell height indicates the original cell height as opposed to the cell height in **mLEF**.

Algorithm 1 DP-based partitioning.

1: calculate $cost(x^l, y^b, x^r, y^t, 0)$ $\forall X^l \leq x^l \leq x^r \leq X^r, Y^b \leq y^b \leq y^t \leq Y^t$
2: **for** $k := 1$ to K **do**
3: **for** $x^l := X^l$ to $X^r - \Delta x$ **do**
4: **for** $y^b := Y^b$ to $Y^t - \Delta y$ **do**
5: **for** $x^r := x^l + \Delta x$ to X^r **do**
6: **for** $y^t := y^b + \Delta y$ to Y^t **do**
7: $cost(x^l, y^b, x^r, y^t, k) = \min_{x^l \leq x \leq x^u, y^b \leq y \leq y^t} ($
$cost(x^l, y^b, x, y^t, k') + cost(x, y^b, x^r, y^t, k'') + 4 \cdot w_{site} \cdot (y^t - y^b),$
$cost(x^l, y^b, x^r, y, k') + cost(x^l, y, x^r, y^t, k'') + d \cdot (x^r - x^l)$
$) \quad \forall k',\ k''\ s.t.\ k' + k'' = k - 1$
8: **end for**
9: **end for**
10: **end for**
11: **end for**
12: **if** $cost(X^l, Y^b, X^r, Y^t, k) \geq cost(X^l, Y^b, X^r, Y^t, k-1)$ **then**
13: **return** $cost(X^l, Y^b, X^r, Y^t, k-1)$
14: **end if**
15: **end for**
16: **return** $cost(X^l, Y^b, X^r, Y^t, K)$

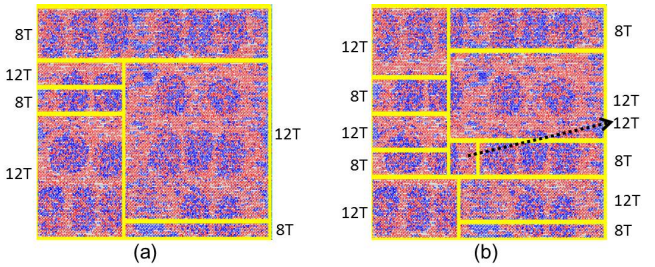


Fig. 5: Examples of partitioning solutions for the AES testcase. In red are 12T cells (with **mLEF**); and in blue are 8T cells. Yellow lines are cuts. The cell height of a partition is marked on its side. (a) Cut number = 5, cost = 4818μm^2. (b) Cut number = 10, cost = 4584μm^2.

The heart of the dynamic programming recurrence (i.e., in determining the partitioning solution with minimum cost) is given in Lines 2-16. We recursively search for the minimum-cost partitioning solution of a rectangular region with k cuts, and increase the value of k in each iteration up to a given maximum allowable number of cuts, K, which is a user-defined parameter.[7] To find the best partitioning solution of a region (x^l, y^b, x^r, y^t) using exactly k cuts, we observe that such a solution can always be seen as a single "top-level" cut, along with the best solutions of the two sub-regions induced by that cut. Hence, to find the best k-cut solution, we enumerate all potential vertical and horizontal cuts of the region, and select the solution that minimizes the sum of the costs of the two separate parts (sub-regions) – with respective number of cuts k' and k'' satisfying $k' + k'' = k - 1$ – plus the cost of the single vertical or horizontal "top-level" cut. Note that the proposed partitioning comprehends the area cost of breaker cells, for which width $= 4 \cdot w_{site}$ for a vertical cut, and height $= d$ for a horizontal cut (Line 7). For the example shown in Figure 3, d must be larger than 64nm. The procedure terminates when the cost does not decrease with an increased cut number (Line 13), or the maximum cut number K is achieved (Line 16). To improve the scalability, we divide the block area into $M \times N$ grids (where M and N are also user-defined parameters), and perform the proposed partitioning method on these grids. The runtime complexity of the procedure is $O((M+N)(M \cdot N \cdot K)^2)$.[8]

[7]In our experiments, we set K to a large value (e.g., 30 for a 100μm×100μm floorplan) in order to ensure good solution quality.

[8]In our experiments, partitioning with number of grids no larger than 30 × 30, and maximum cut number no larger than 40, requires less than one minute of a single thread on a 2.5GHz Intel Xeon server.

B. Timing-Aware Placement Legalization

Based on the partitioning solution, we perform iterative optimization to achieve a legal placement. Note that we still use **mLEF** at this optimization stage, but boundaries and cell heights of regions have been defined. We apply two knobs in our iterative heuristic: *displacement of a cell* (e.g., moving a 12T cell from an 8T region to a 12T region), and *cell-height swapping* (e.g., assign an 8T cell to a 12T cell master in a 12T region via gate sizing). Both of these knobs affect timing, and cell-height swapping also affects area. Thus, to ensure that the optimization does not lead to large design quality degradation, we evaluate the timing and area impacts of each potential move (one move is a cell displacement or a cell-height swap).

Because timing analysis with commercial P&R tools is typically slow, our optimization approach requires a relatively accurate and fast timing engine. We have developed an internal timing analysis engine (i.e., *internal timer*) to guide the optimization. Our internal timer estimates gate delay and slew at an output pin based on the Liberty lookup tables. It further uses D2M [1] and PERI [9] models that respectively estimate wire delay and slew propagation along the interconnect. Wirelength change due to cell displacement is measured by net HPWL (Half-Perimeter Wire Length), and wire capacitance and resistance are scaled correspondingly.

To comprehend wire congestion effects, we add a penalty in the form of wire resistance and capacitance scaling, based on routing demand vs. supply overflows within the bounding box of a given net.[9] More specifically, if the average horizontal (resp. vertical) routing congestion within the bounding box of a net is $X\%$, we penalize the horizontal (resp. vertical) portion of HPWL by a multiplicative factor of ($X\%$-$X_{th}\%$) whenever $X > X_{th}$. Here, $X_{th}\%$ is a threshold that we set to 95% based on separate studies. The value $X_{th}\% = 95\%$ is used in all experiments reported below.[10] To maintain the accuracy of our internal timer, we correlate timing slack, wire capacitance and overflow information during the optimization through a Tcl socket with *Cadence SoC Encounter* [17]. Figure 6 shows our optimization framework. We believe that our internal timer approach most closely resembles that of the previous work [8]; however, our internal timer better comprehends the impact of cell displacement on timing by considering both wirelength change and routing congestion information.

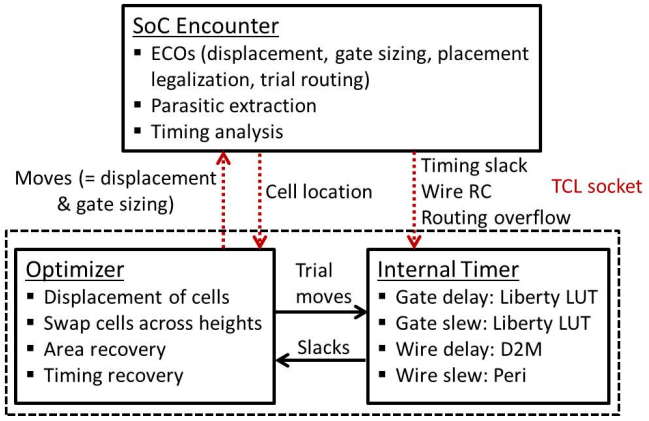


Fig. 6: Framework of our optimization.

Algorithm 2 describes our heuristic to legalize the placement. We first evaluate the cost (in terms of area and timing) of each potential move (i.e., cell displacement or swapping) (Line 4). We consider

Algorithm 2 Heuristic to legalize placement.

```
1: while there exists a cell with different height than its partition do
2:   list ← ∅
3:   for all cell g with different height than its partition do
4:     calculate cost function of g
5:     add g to list
6:   end for
7:   sort list in order of decreasing cost
8:   swap_cnt ← 0
9:   for all g ∈ list do
10:      apply displacement/swapping based on cost function
11:    incremental timing analysis
12:    if slack of g < min(0, original slack of g) || whitespace_of_grid < ω
       then
13:      undo change
14:    else
15:      ++swap_cnt
16:    end if
17:    if swap_cnt ≥ γ · total_gate_count then
18:      apply ECOs in SoC Encounter
19:      correlate internal timer with SoC Encounter
20:      for all cell g in the design do
21:        downsize g
22:        incremental timing analysis
23:        if slack of g < min(0, original slack of g) then
24:          undo change
25:        end if
26:      end for
27:      if WNS ≤ −θ · clock_period then
28:        fix maximum transition violations
29:        timing recovery
30:        apply ECOs in SoC Encounter
31:        correlate internal timer with SoC Encounter
32:      end if
33:    end if
34:  end for
35: end while
```

cell displacement in eight directions (i.e., {N, S, E, W, NE, NW, SE, SW}) with the maximum movement distance of D (D = 15μm in our experiments). The set of candidate cell displacements is similar to what is applied in the local optimization of [6]. For cell-height swapping, we consider candidate library cells whose heights match that of the partition. We use the cost function shown in Equation (3)

$$Cost = \alpha \cdot \frac{\max(0, -\Delta\text{slack})}{\max(1\text{ps}, \text{slack}_{orig})} + (1-\alpha) \cdot \frac{\max(0, \Delta\text{area})}{\max(1\mu\text{m}^2, \text{whitespace}_{orig})} \quad (3)$$

where Δslack and Δarea are respectively the timing slack and cell area changes due to displacement and/or swapping. slack$_{orig}$ and whitespace$_{orig}$ are the original timing slack of the cell and whitespace of the corresponding grid. We divide the block area into an $M \times N$ mesh of grids. For each grid, we estimate whitespace based on placement utilization. The parameter α is a weighting factor, which has an initial value of 0.5. We adaptively change the value of α for each cell during the iterative optimization, such that when an attempt leads to timing violation (resp. placement utilization violation), we increase (resp. decrease) α of the cell by 1.5×.

We sort all cells which have different height than their partition in decreasing order of cost, and apply moves to legalize the placement (Line 7). When a move results in timing failure or violation of placement density, we undo the move (Lines 12-13); here ω is the required whitespace according to the area of breaker cells and maximum placement density constraints.[11] To ensure the convergence of the flow (i.e., that optimization can lead to a legalized placement), we commit the move of a cell which has been visited F times, regardless of its impact on timing and area.

[9] We estimate overflow based on the trial routing solution from *Cadence SoC Encounter* [17].

[10] For example, if the average horizontal congestion is 98%, we multiply the x-component of HPWL by 1.03 = 0.98 / 0.95.

[11] In our experiments, we set the maximum placement density of the entire block as the placement density from the initial placement plus 5%.

We use $F = 5$ in our optimization.[12] In addition, we apply a form of Tabu search [4] during the optimization to increase the likelihood of finding feasible solutions for cells. Specifically, we record the latest three attempts and forbid these moves for the current move of optimization. During the optimization, we (re-)correlate our internal timer with *SoC Encounter* in terms of timing slack/slew, cell location, wire parasitic, and routing overflow after every $\gamma\%$ of the total number of cells has been changed (Lines 17-19). We use $\gamma = 2$ in our optimization. We also include area recovery (Lines 20-26) and timing recovery (Lines 27-32) in our optimization to maintain timing and area quality. The parameter θ is a threshold of slack violation that triggers timing recovery; we empirically set this to 0.15. Note that during the timing recovery, we perform backward (in which we downsize fanout cells) and forward (in which we upsize cells) maximum transition violation fixes, which enhance the timing recovery quality.

We observe from our experimental results that the ratio between the number of cells being swapped and the number of cells being displaced ranges from 1.2 to 5.4. This ratio seems highly dependent on the partitioning solution, timing constraints, netlist structure, etc. For instance, fewer partitions and/or tighter timing constraints can lead to more swaps relative to displacements.

C. Mapping from mLEF to Original LEF in Assigned Regions

As discussed above, during the initial placement, partitioning and legalization stages, we use **mLEF** with adjusted aspect ratio for cell layouts, such that a cell originally with large height becomes shorter and wider. When the placement solution is legalized, we update the floorplan to have cell rows according to the height of each partition. We also allocate space to model the area cost of breaker cells. To map cells to the updated cell rows, we recover the original aspect ratio of cell layouts. For example, assume that there are 20 10T cells uniformly placed on five 8T cell rows (i.e., as a 5×4 mesh). To update the floorplan, we maintain the same partition area and place cell rows according to the height of the partition. We therefore have four 10T cell rows. Given that the layout of these 10T cells (with the same cell area) are scaled back to their original height with a reduced cell width, five cells now can fit into one row in the updated floorplan. The mapped cell placement becomes a 4×5 mesh. As shown in the example, cell mapping in the updated floorplan can be viewed as embedding a graph to another graph with a different aspect ratio (e.g., embed a 5×4 mesh to a 4×5 mesh). We therefore revisit the graph embedding literature.

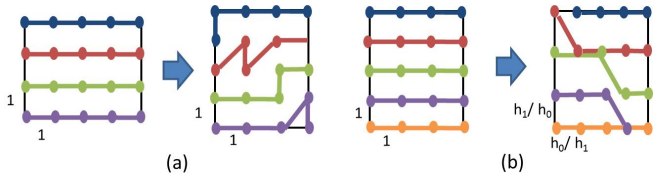


Fig. 7: Illustration of graph embedding (a) from [3], and (b) for proposed cell mapping. Vertical connections are not shown.

Ellis [3] shows that to embed a 2D mesh of size $w \times h$ (with unit distance between every two adjacent nodes in both horizontal and vertical directions) to another 2D mesh of size $w' \times h'$, where $w' < w$ and h' is the smallest integer satisfying $w' \cdot h' \geq w \cdot h$, if $\frac{w}{w'}$ is no larger than 2, the maximum wirelength of a two-pin net (in Manhattan distance) in the embedded graph is no more than two units. An example with $\frac{w}{w'} = \frac{5}{4}$ is shown in Figure 7(a): the wirelength of each connection in the original graph is one, and the maximum wirelength in the embedded graph (i.e., the diagonal connection) is two. Note that our optimization of cell mapping differs from [3], in that [3] varies the area of the graph (i.e., mesh) while our optimization assumes a fixed mesh area (i.e., area of a partition).

[12]We observe in our experiments that the number of cells which have been visited six times without a feasible solution is quite small, e.g., less than 60 in a design with 15K cells.

Algorithm 3 Cell mapping.

1: $W_{avg} = (\sum_{g \in P_j} w_g) \ / R'$
2: $i = 1$
3: $list \leftarrow$ cells on i^{th} row of original floorplan
4: sort $list$ in order of increasing cell width
5: **for** $i' := 1$ to R' **do**
6: $\quad list' \leftarrow \emptyset$; $\quad W = 0$
7: $\quad$ **while** $(i' == R') \ || \ (W < W_{avg})$ **do**
8: $\quad\quad g \leftarrow list.pop\_front()$
9: $\quad\quad list'.push(g)$
10: $\quad\quad W$ += w_g
11: $\quad\quad$ **if** $list == \emptyset$ **then**
12: $\quad\quad\quad ++i$
13: $\quad\quad\quad$ **if** $i == (R + 1)$ **then**
14: $\quad\quad\quad\quad$ **return**
15: $\quad\quad\quad$ **end if**
16: $\quad\quad\quad list \leftarrow$ cells on i^{th} row of original floorplan
17: $\quad\quad\quad$ sort $list$ in order of increasing cell width
18: $\quad\quad$ **end if**
19: $\quad$ **end while**
20: $\quad$ place cells in $list'$ on i'^{th} row of updated floorplan
21: $\quad$ legalize cell placement on i'^{th} row
22: **end for**

Following the discussions in [3], we can show that if we map a 2D-mesh placement with cell height h_0, in which all cells have the same cell area, to another 2D-mesh placement with cell height h_1, the maximum wirelength scaling of a mesh edge (i.e., two-pin net) according to the mapping is no more than $\frac{h_0}{h_1} + \frac{h_1}{h_0}$.[13] Figure 7(b) shows an example with 10T and 8T cells. Assuming unit wirelength for each two-pin connection between any horizontally or vertically adjacent cells in the original 2D-mesh placement, the maximum wirelength increase is 1.05.

Inspired by the graph-embedding theory, we propose a method to map cells onto cell rows with recovered cell heights for general cases, in which cells can have different widths and are not necessarily placed as a 2D mesh. Algorithm 3 shows our procedure to map cells from an original floorplan with R rows of height h_0 to an updated floorplan with R' rows of height h_j. Our method is similar to legalization approaches such as those in [7] and [10] in that we sort cells first and then legalize one at a time. We first estimate the average total cell width of each row in the updated floorplan (Line 1), in which g is a cell in partition P_j; w_g is the actual width of the cell corresponding to height h_j. We then sort the cells on the i^{th} row of the initial floorplan by increasing widths (Line 4).[14] We iteratively assign cells to the i'^{th} row of the updated floorplan (Lines 5-22). When all cells from the i^{th} row of the initial floorplan are mapped, we collect cells from the $(i+1)^{th}$ row (Lines 11-18). When the total width of the mapped cells exceeds the average total width of each row (i.e., W_{avg}), we place the selected cells on the i'^{th} row of the updated floorplan with their original X-coordinates (Line 20). Finally, we legalize the placed cells in each row with awareness of area of breaker cells by traversing the cells from left to right, and from right to left (Line 21). It is obvious that the proposed algorithm can achieve the mapping solution shown in Figure 7(b), given appropriate tie-breaking, i.e., when sorting cells with the same width (Lines 4, 17), we prioritize the one with a smaller X-coordinate value; when legalizing placement of cells with the same X-coordinate (Line 21), we prioritize those originally located on even-numbered rows.

[13]Proof details are given in [3].

[14]Our separate studies of different sorting criteria (X-coordinate, cell width, overlap with placed cells, etc.) find that sorting by cell width leads to smallest perturbations of cell placement, and achieves the smallest wirelength. Further, we observe in our experiments that the average wirelength increase due to cell mapping, taken over 23 designs, is only 0.8%.

V. Experimental Results

We perform experiments in a 28nm LP foundry technology with dual-VT libraries, 0.95V nominal supply voltage, and cell height choices 12T and 8T. To confirm that our optimization can perform a fine-grained mixed cell-height implementation, we select four design blocks (AES, DES, DMA, MPEG) from the OpenCores [14] website. Parameters of these four testcases are shown in Table II. For each design, we determine a range of clock periods starting from a clock period with relative loose timing constraint, up to the clock period at which the 8T-only implementation shows setup timing violations. These designs are synthesized using *Synopsys Design Compiler vH-2013.03-SP3* [15] and then placed and routed using *Cadence SoC Encounter vEDI14.1* [17]. We set the gate density at the floorplan stage as 60%. We respectively use *Cadence SoC Encounter* and *Synopsys PrimeTime-PX vH-2013.06-SP2* [16] for timing and power analysis at the post-routing stage (with ideal clocks) and wire parasitics (SPEF) obtained from SoC Encounter. We use *Synopsys PrimeTime vH-2013.06-SP2* [16] to search for the minimum supply voltage that satisfies a given frequency target. Our optimization flow is implemented in C++. Functions used in P&R tools and the socket between our optimizer and the P&R tool are implemented in Tcl. We conduct our experiments on a 2.5GHz Intel Xeon server.

TABLE II: Benchmarks.

Design	#Instances	#Flip-flops	Clock period range
AES	∼15K	530	700ps – 1ns
DES	∼22K	1984	650ps – 800ps
DMA	∼1.5K	277	350ps – 500ps
MPEG	∼13K	3193	600ps – 750ps

Modeling breaker cell costs. The placement site pitch (width) and the M2 metal pitch in the 28nm LP technology that we use are respectively 0.136μm and 0.1μm. Based on the discussion in Section III, the horizontal and vertical shifts between any 8T and 12T regions must be no less than 0.544μm and 0.1μm, respectively. In addition, to preserve cell row alignment in the design, we shift cell rows by 0.8μm in the vertical direction between any 12T and 8T regions. We also insert placement and routing blockages correspondingly. Figure 8 shows one layout example.

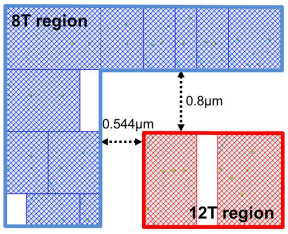


Fig. 8: Inserted space on the boundaries between 12T and 8T regions to model the cost of breaker cells.

A. Comparison at V_{nom}

We implement our benchmark designs using our proposed flow with mixed 8T/12T cells. We also perform conventional SP&R (synthesis, placement and routing) with 12T-only cells and 8T-only cells for comparison. The designs are implemented with clock periods shown in Table II. We use the nominal voltage 0.95V at (SS, 125°C) corner for design implementation and timing analysis. We further use corner (TT, 1.05V, 25°C) for leakage power analysis. Total power values are reported at the signoff frequency. We divide the block area of each design into grids of size around 6μm × 6μm for partitioning and placement density evaluation. Parameters of the implemented designs are described in Table III, in which the clock period is the clock period used for implementation. Figure 9 further shows the Pareto curves illustrating tradeoffs between area and performance of implemented designs at the post-routing stage, where the frequency given is the maximum achievable operating frequency. Our experiments show that with loose timing constraints, 8T-only implementation has minimum design area, and 12T-only implementation has large area overhead. On the other hand, the maximum achievable performance of an 8T-only design is limited by the weak drive strengths of 8T cells. Designs with mixed cell heights achieve significant (i.e., 20%) area reduction on average as compared to 12T-only designs, while maintaining similar performance (i.e., post-routing slack differences that are less than 40ps). Moreover, mixed cell-height design improves performance significantly (e.g., by 20% for design AES) over the 8T-only design with the same total cell area.[15] However, we also observe large wirelength increase for certain optimized designs (e.g., DMA). This is because our cost function for displacement of cells only considers timing penalty but not its impact on wirelength. Understanding the effects of legalization on wirelength and corresponding power overhead is among our future works.

B. Comparison with Voltage Scaling

Given that certain designs have timing violations, to achieve a fair power comparison we perform voltage scaling on each design so that all designs meet the timing constraints. We then compare power at the scaled supply voltage. In our experiments, we define *scaling_lib_group* in the *PT-PX* tool to enable such comparisons. Note that to compensate the slack discrepancy between *SoC Encounter* and *PrimeTime*, we apply a constant slack shift of the entire block to correlate the post-routing worst slack values, then perform voltage scaling. When the difference between the scaled voltage and the signoff voltage is larger than 30mV, we perform SP&R with the scaled voltage and use the smaller power value between that of the initial implementation and that of the additional implementation in our comparison. Figure 10 shows the power comparison. We observe that designs implemented with 8T-only, 12T-only and mixed cell heights in general have similar total power. A possible explanation: although 12T cells have larger capacitance, an 8T-only implementation tends to have a larger number of instances, thus leading to similar total capacitance and power consumption. The design MPEG implemented with 12T-only cells shows larger power, apparently due to its large number of flip-flops, which have high toggle rates. We note that the relatively high power of the optimized designs (e.g., AES, DES, DMA) with mixed cell heights is likely due to our cost function (i.e., Equation (3)) being unaware of power consumption. This is the subject of current investigations.

VI. Conclusion and Future Works

In this work, we have proposed a novel physical design optimization flow to mix cells with different heights in a fine-grained manner within a single place-and-route block. Our flow addresses the "chicken-and-egg" loop between floorplan site definition and the post-placement choice of cell heights, and correctly models (based on industry feedback from 20SOC and 16FF design experience) "breaker cell" overheads of the mixed-height placement. Our optimization, applied to production 12T and 8T libraries in a 28LP foundry technology, can achieve 25% area reduction, while maintaining performance, as compared to a 12T-only design flow. Moreover, our optimized mixed-height designs can achieve significant performance increase as compared to designs with 8T-only cells.

We observe from our results that the mixed cell-height implementation can, for certain testcases, have relatively larger values of power and wirelength as compared to the 12T-only and

[15]We recognize that the larger number of instances in the 12T-only implementation of AES is unexpected. We believe that this is partly due to the 12T implementation's superior performance as compared to the 8T-only and mixed-height implementations. The 12T-only AES implementation also has more small-size buffers (e.g., 488 more than the mixed-height implementation), which may ultimately stem from having different post-synthesis netlists.

TABLE III: Parameters and results of implemented designs.

Design (Clock period)	AES (700ps)			DES (650ps)			DMA (350ps)			MPEG (600ps)		
Flow	12T	8T	mix	12T	8T	mix	12T	8T	mix	12T	8T	mix
#Instances	16006	14441	15544	21878	23503	23360	1554	1735	1667	12548	14609	14524
12T/8T	16006/0	0/14441	11799/3745	21878/0	0/23503	20204/3156	1554/0	0/1735	1339/328	12548/0	0/14609	2425/12099
LVT/RVT	8110/7896	13168/1273	12171/3373	8009/13869	21146/2357	10791/12569	1102/452	1517/218	1134/533	5983/6565	9992/4617	7296/7228
Setup WNS (ps)	0	-94	-37	0	0	0	-4	-90	-17	3	-34	-12
Setup TNS (ns)	0	-12.42	-4.57	0	0	0	-0.03	-14.83	-0.27	3	-2.47	-0.09
#Hold violations	0	0	0	0	0	0	0	0	0	0	0	0
Area (μm^2)	17169	12966	11097	23328	21913	16816	2445	1975	1837	20386	14967	14789
Utilization	77%	95%	65%	69%	86%	51%	79%	75%	54%	63%	69%	59%
WL (μm)	190514	152437	214240	200573	197147	255468	14614	16205	20657	219516	150520	183450
Leakage power (mW)	0.086	0.081	0.119	0.065	0.114	0.088	0.012	0.009	0.014	0.054	0.049	0.048
Total power (mW)	34.2	27.3	32.1	53.6	56.5	57.2	4.59	3.84	4.74	23.9	20.0	20.3
Runtime (min)	48	58	149	78	77	132	14	9	17	13	27	21

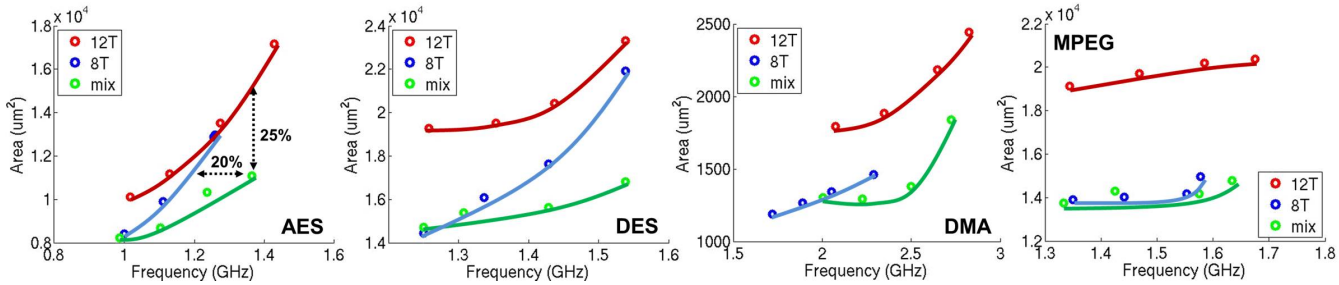


Fig. 9: Pareto curves of performance-area tradeoff for implementations with 8T-only, 12T-only and mixed cells.

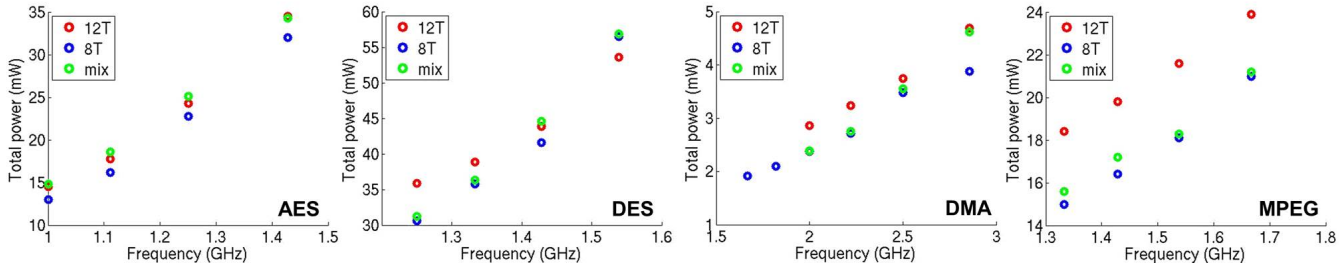


Fig. 10: Iso-performance power comparison with voltage scaling among implementations with 8T-only, 12T-only and mixed cells.

8T-only implementations. We believe that this is because our cost function only considers timing and area during the optimization. A clear direction for future work is to better comprehend power and wirelength costs in our optimization. Other future and ongoing works include: (i) a clock tree synthesis flow with mixed cell heights; (ii) a more comprehensive cost function that can trade off performance, power, area and wirelength in guiding the optimization; (iii) more holistic understanding of the interactions among partitioning, cell displacement and cell-height swapping; (iv) an iso-utilization comparison between our optimization and single cell-height implementations; (v) improved logic synthesis (e.g., constraints methodologies or target library models) for the mixed-height regime; and (vi) mitigation of routing congestion and pin accessibility issues (e.g., by mixing in a small portion of cells with larger height).

ACKNOWLEDGMENTS

We thank Ms. Nancy MacDonald for valuable feedback and discussions during the course of our project.

REFERENCES

[1] C. J. Alpert, A. Devgan and C. Kashyap, "A Two Moment RC Delay Metric for Performance Optimization", *Proc. ISPD*, 2000, pp. 73-78.
[2] R. L. S. Ching, E. F. Y. Young, K. C. K. Leung and C. Chu, "Post-Placement Voltage Island Generation", *Proc. ICCAD*, 2006, pp. 641-646.
[3] J. A. Ellis, "Embedding Rectangular Grids Into Square Grids", *IEEE Trans. Computers* 40(1) (1991), pp. 46-51.
[4] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1999.
[5] L. Guo, Y. Cai, Q. Zhou and X. Hong, "Logic and Layout Aware Voltage Island Generation for Low Power Design", *Proc. ASP-DAC*, 2007, pp. 666-671.
[6] K. Han, A. B. Kahng, J. Lee, J. Li and S. Nath, "A Global-Local Optimization Framework for Simultaneous Multi-Mode Multi-Corner Clock Skew Variation Reduction", *Proc. DAC*, pp. 26:1-26:6, 2015.
[7] D. Hill, "Method and System for High Speed Detailed Placement of Cells Within an Integrated Circuit Design", *US Patent 6370673*, April 2002.
[8] A. B. Kahng, S. Kang, H. Lee, I. L. Markov and P. Thapar, "High-Performance Gate Sizing with a Signoff Timer", *Proc. ICCAD*, 2013, pp. 450-457.
[9] C. V. Kashyap, C. J. Alpert, F. Liu and A. Devgan, "PERI: A Technique for Extending Delay and Slew Metrics to Ramp Inputs", *Proc. TAU*, 2002, pp. 57-62.
[10] P. Spindler, U. Schlichtmann and F. M. Johannes, "Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement", *Proc. ISPD*, 2008, pp. 47-53.
[11] H. Wu and M. D. F. Wong, "Improving Voltage Assignment by Outlier Detection and Incremental Placement", *Proc. DAC*, 2007, pp. 459-464.
[12] H. Wu, I.-M. Liu, M. D. F. Wong and Y. Wang, "Post-Placement Voltage Island Generation under Performance Requirement", *Proc. ICCAD*, 2005, pp. 309-316.
[13] H. Wu, M. D. F. Wong and I.-M. Liu, "Timing-Constrained and Voltage-Island-Aware Voltage Assignment", *Proc. DAC*, 2006, pp. 429-432.
[14] *OpenCores. http://opencores.org*
[15] *Synopsys Design Compiler User's Manual.*
[16] *Synopsys PrimeTime User's Manual.*
[17] *Cadence SOC Encounter User Guide.*

APPENDIX

We list all the user-defined parameters of our optimizer in Table IV.

TABLE IV: User-defined parameters.

Term	Meaning
K	maximum number of cuts
$M \times N$	number of grids
D	maximum displacement distance
γ	determines correlation frequency
θ	slack violation tolerance threshold
F	maximum number of visits of a cell before a move is applied