HybridGP: Global Placement for Hybrid-Row-Height Designs*

Kuan-Yu Chen, Hsiu-Chu Hsu, Wai-Kei Mak, Ting-Chi Wang

Department of Computer Science, National Tsing Hua University, Taiwan
kychen1013@gapp.nthu.edu.tw, chieya.ch@gmail.com, wkmak@cs.nthu.edu.tw, tcwang@cs.nthu.edu.tw

2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC) | 978-1-6654-2135-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASP-DAC52403.2022.9712565

***Abstract*—Conventional global placement algorithms typically assume that all cell rows in a design have the same height. Nevertheless, a design making use of standard cells with short-row height, tall-row height, and double-row (short plus tall) height can provide a better sweet spot for performance and area co-optimization in advanced nodes. In this paper, we assume for a hybrid-row-height design, its placement region is composed of both tall rows and short rows, and a cell library containing multiple versions of each cell in the design is provided. We present a new analytical global placer, HybridGP, for such hybrid-row-height designs. Furthermore, we assume that a subset of cells with sufficient timing slacks is given so that we may change their versions without overall timing degradation if desired. Our approach considers the usage of short-row and tall-row resources and exploits the flexibility of cell version change to facilitate the subsequent legalization stage. Augmented with an identical legalizer for final placement legalization, we compared HybridGP with a conventional global placer. The experimental results show that legalized placement solutions of much better quality can be obtained in less run time with HybridGP.**

I. Introduction

Standard cells traditionally have the same height (which is equal to the row height), but as the diversified demands increase, the multi-row-height cell structure has been proposed [1] where the height of a cell is an integral multiple of the row height. Another design strategy is to mix the usage of standard cells of different heights, where the height of a taller cell (e.g., a 12-track cell) is no longer an integral multiple of the height of a shorter cell (e.g., a 8-track cell) [4], [11], [3]. For such designs, a layout is divided into regions such that each region can accommodate only cells of the same height and therefore could have a different row height from another region. However, area overhead is incurred as it needs to reserve vertical or horizontal spacing between neighboring regions of different row heights for avoiding the P/G rail encroachment and other issues.

As technology advances, another kind of hybrid-row-height designs has been recently introduced [5], [8]. For such designs, a layout is partitioned into rows of two different heights, which are respectively called *tall rows* and *short rows* in this paper. The tall-row height is not an integral multiple of the short-row height. Moreover, no spacing is required between two adjacent rows (i.e., two adjacent rows can abut just like those in uniform-row-height designs). See Fig. 1 for an example. Compared with uniform-row-height designs, such new and advanced hybrid-row-height designs can provide a better sweet spot for performance and area co-optimization. Besides, they do not have any spacing overhead compared with the other kind of hybrid-row-height designs mentioned in the last paragraph.

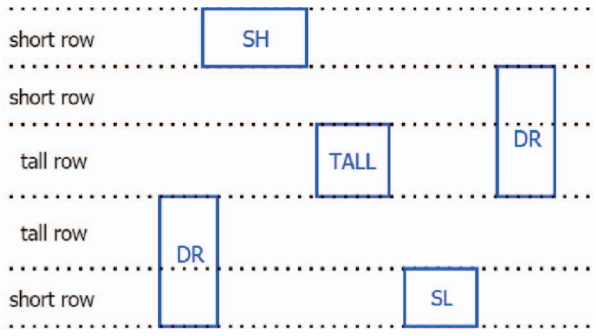

Fig. 1. Row configuration and cell versions.

In this paper, we study a global placement problem for advanced hybrid-row-height designs and it has never been addressed in the literature before. The problem is described as follows. A synthesized netlist $N = (V, E)$, a cell library, and a placement region composed of short and tall rows are given. The vertex set V of N contains three objects, namely movable cells, pre-placed macros and I/O pins, while E is the net set. The x and y coordinates of I/O pins and macros are fixed and unchangeable. To well utilize the given tall-row and short-row resources to enhance the chance of finding a legal placement later in the legalization stage, a subset of movable cells in V with sufficient timing slacks is specified so that during global placement we may change their versions (according to the substitution rules to be introduced soon) without overall timing degradation if desired. We aim to find out the x and y coordinates and the version of each movable cell such that the total half-perimeter wirelength (HPWL) after the legalization stage is as small as possible.

For the given cell library, we have the following assumptions. There are four versions for each cell: SL (short-row height and low-driving strength), SH (short-row height and high-driving strength), TALL (tall-row height and high-driving strength), and DR (double-row height and high-driving strength) [8]. For each cell, its SH version is wider

*This work was supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 107-2221-E-007-081-MY3 and MOST 110-2221-E-007-122.

than its SL version, and its TALL version is wider than its DR version but thinner than its SH version. The DR version of each cell has the height equal to the sum of the tall-row height and the short-row height. In addition, to facilitate the alignment of P/G rails, the DR version of each cell has two implementations of the same size: One can be placed on a row pair with the tall row on top, while the other can be placed on a row pair with the short row on top. See Fig. 1 for an example.

For each cell, we assume that each of its high-driving versions has a smaller delay but a larger area than its low-driving version [8]. Therefore for each cell in the specified subset of cells that have sufficient timing slacks and hence are allowed to change versions during global placement, our substitution rules are as follows: (i) If it is an SL version, it can remain unchanged or can be converted to its SH, TALL, or DR version. (ii) If it is a high-driving version (i.e., SH, TALL, or DR), it can remain unchanged or can be converted to one of the other two high-driving versions.

In this work, we present an analytical global placement approach, HybridGP, for advanced hybrid-row-height designs. It is based on the quadratic technique with the concept of anchor for cell spreading. The contributions of this paper are summarized as follows:

1) To the best of our knowledge, this is the first work in the literature that can handle global placement for advanced hybrid-row-height designs.
2) Effective methods are proposed and embedded into HybridGP to find local overflow bins, spread cells, and change cell versions.
3) The experimental results show that by augmenting an identical legalizer, HybridGP can help produce a legalized placement of much better quality with less run time when compared with a conventional global placer.

The rest of this paper is organized as follows. In Sec. II, we provide an overview of our approach. We detail our approach in Sec. III and show the experimental results in Sec. IV. In Sec. V, we conclude the paper.

II. Overview of Our Approach

In quadratic placement, all nets are assumed to be two-pin nets and the HPWL is replaced by the quadratic wirelength which is calculated by squared Euclidean distance between two pins of each net. The quadratic wirelength can be then minimized for the x-coordinates and y-coordinates, respectively, by solving two linear systems. Because the cost function only focuses on quadratic wirelength, quadratic placement demands additional techniques for cell spreading. A popular cell spreading method is based on the concept of anchor. Typically, a quadratic placer, e.g., [7], iteratively solves linear systems, finds an anchor for each cell, and adds a pseudo net connecting the cell and anchor until the solution meets the convergence condition.

Our global placer HybridGP is a quadratic placer based on the above-mentioned idea, and its overall flow is shown in Fig. 2. It consists of two stages: initial stage and refined stage. In the initial stage, it uses two kinds of net model to

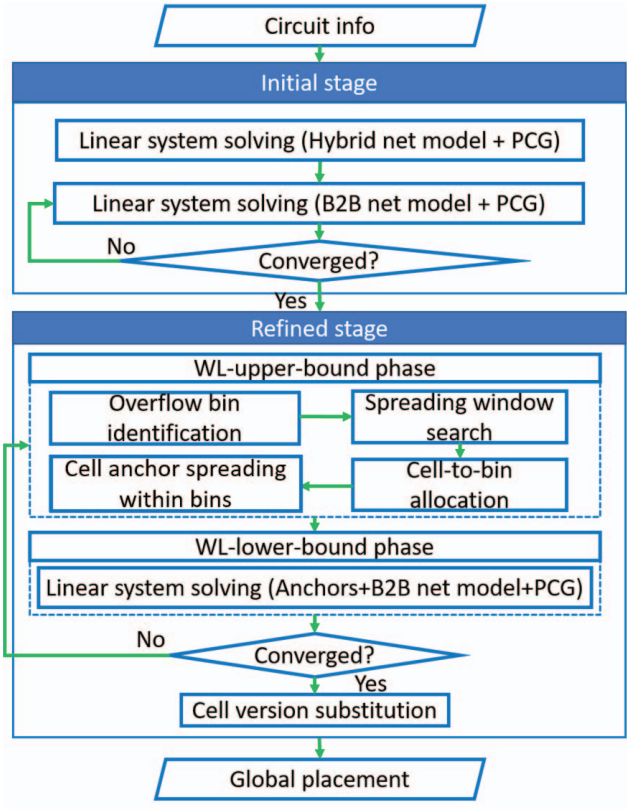

Fig. 2. The overall flow of HybridGP.

decompose each net with 3 or more pins into a set of 2-pin nets. The hybrid net model [10] is first adopted and the two linear systems are solved. Next, it uses the B2B model [9] and solves the two linear systems; this step is repeated a few times (four times in our current implementation) to obtain a converged wirelength. Each linear system is solved by the preconditioned conjugate gradient (PCG) method with incomplete Cholesky decomposition.

In the refined stage, HybridGP iteratively performs the WL-upper-bound phase and the WL-lower-bound phase until the wirelength difference between the placements generated by the two phases is within a user-specified bound (less than 3% in our current implementation). In the WL-upper-bound phase, cells are spread and their anchors are updated. This phase is similar to a rough legalization and hence will increase the wirelength. In the WL-lower-bound phase, the B2B model is adopted and a pseudo net connecting each cell and its anchor is added to the net set. Then the two linear systems are solved to find the new location of each cell, which will produce wirelength less than that from the WL-upper-bound phase.

The WL-upper-bound phase is composed of four steps: overflow bin identification, spreading window search, cell-to-bin allocation, and cell anchor spreading within bins. Firstly, all overflow bins in the current placement solution are identified. Secondly, it finds a window with enough resources for spreading the cells of each overflow bin. Next, it assigns each movable cell to a suitable bin within a window. Finally,

it determines the suitable position of each movable cell in each bin. After the WL-upper-bound phase, the new positions of movable cells are treated as the anchors and updated in the two linear systems.

After the wirelength gap between the two phases is converged, HybridGp proceeds to perform cell version substitution, and completes the refined stage. More details of HybridGP are described in the next section.

III. Details of Our Approach

A. Overflow Bin Identification

In the WL-upper-bound phase, our main goal is to drastically reduce the resource overflow of bins after the WL-lower-bound phase. We divide the layout into m × n bins, and then we predict which bins are definite overflow bins. The way we decide the bin size is similar to [7] but a bin may span a combination of short rows and tall rows.

Our method of determining if a bin is actually an overflow bin is different from all previous works. Firstly, for a hybrid-row-height design, if the resource usage of either short or tall rows in a bin exceeds that provided by the bin, then it is an overflow bin even if the total area of all cells placed in the bin is less than the available placement area of the bin. Secondly, we also consider the flexibility of cell version change of a subset of cells with sufficient timing slacks to judge whether a bin will overflow or not. For simplicity, we call the cells with sufficient timing slacks the *flexible cells*. If cell version substitution of the flexible cells can help make both the resource usage of short rows and the resource usage of tall rows in the bin not to exceed those provided by the bin, then we will deem this bin as a non-overflow bin. Our method to determine a bin will overflow or not is given in Algorithm 1. Note that we do not actually change the version of any cell at this step because cells may still be moved around subsequently. We will only perform cell version change of flexible cells once at the last step of the global placement process.

Algorithm 1: Hybrid-row-height overflow prediction

Input: A bin b. The cell list in b. User-defined density limit UDL. t = average ratio of a cell's TALL version width to its SL version width. s = average ratio of a cell's SH version width to its TALL version width.
Output: Bin b is predicted to overflow or not.

```
1  W_T = total available width of tall rows in b;
2  W_S = total available width of short rows in b;
3  X_1 = current usage of short row resource by all non-flexible cells
     and the SL version of all low-driving flexible cells in b;
4  X_1^* = sum of the SL version width of all low-driving flexible cells
     in b;
5  X_2 = current usage of tall row resource by all non-flexible cells
     and the TALL version of all high-driving flexible cells in b;
6  X_2^* = sum of the TALL version width of all high-driving flexible
     cells in b;
7  if X_1 > W_s × UDL and X_2 > W_t × UDL then
8      overflow = true
9  if X_1 ≤ W_s × UDL and X_2 ≤ W_t × UDL then
10     overflow = false
11 if X_1 > W_s × UDL and X_2 ≤ W_t × UDL then
12     if X_1 − X_1^* > W_s × UDL or
         (X_1 − W_s × UDL) × t > (W_t × UDL − X_2) then
13         overflow = true
14     else
15         overflow = false
16 if x1 ≤ W_s × UDL and X_2 > W_t × UDL then
17     if X_2 − X_2^* > W_s × UDL or
         (X_2 − W_t × UDL) × s > (W_s × UDL − X_1) then
18         overflow = true
19     else
20         overflow = false
```

In Algorithm 1, lines 3-6 first treat all low-driving flexible cells as SL versions and all high-driving flexible cells as TALL versions to estimate the resource usage since a low-driving cell can be in any of the four version but SL version is the most area efficient, and similarly a high-driving cell can be in TALL/SH/DR version but TALL version is the most area efficient among the three. Line 7 checks the utilization of tall-row resource and short-row resource under the above assumption. If both resources are insufficient, then b is predicted as an overflow bin since no cell version substitution is going to help. Line 9 checks whether both resources are sufficient under the same assumption. If so, b is predicted as a non-overflow bin. In lines 11-15, when the short-row demand may be too high but the tall-row demand is not, we check if converting some short-row demand of flexible cells into tall-row demand will be able to avoid overflow or not. If yes, this bin is predicted as a non-overflow bin, otherwise it is predicted as an overflow bin. Similarly, in lines 11-15, when the tall-row demand may be too high but the short-row demand is not, we check if converting some tall-row demand of flexible cells into short-row demand will be able to avoid overflow or not. If yes, this bin is predicted as a non-overflow bin, otherwise it is predicted as an overflow bin.

When a bin is deemed as an overflow bin by Algorithm 1, we will also calculate its degree of overflow as follows. We note the insufficient width of each resource based on the initial version of each cell and we take advantage of the flexible cells to minimize the maximum of the insufficient widths of the two resources. And we use the final maximum of the insufficient widths of the two resources to measure the degree of overflow. But we will not change the versions of the flexible cells immediately. We will wait until the last step and change their versions once and for all.

B. Spreading Window Search

After identifying all the overflow bins, we will cluster neighboring overflow bins into groups of at most three overflow bins each. We refer to each such cluster as a *hotspot*. Then, we eliminate these hotspots in the order of their degrees of overflow. For each hotspot, we find a suitable rectangular window that covers the corresponding hotspot entirely such that all the cells within the window can be relocated within it to resolve local overflow as described next.

For a hotspot, we first find out the most serious overflow bin belonging to it and treat it as the window center. We start from the center and search outwards, and each time we expand outward by one bin width or height. We use

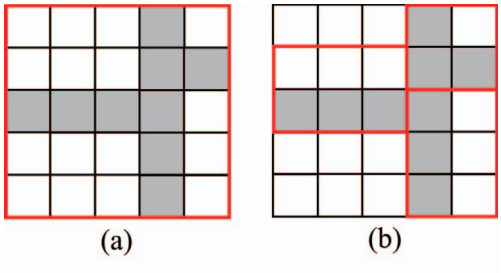

Fig. 3. Hotspot sizes. (a) All neighboring overflow bins treated as one hotspot can require a large rectangular spreading window. (b) Dividing into smaller hotspots will require rectangular spreading windows much smaller in size.

a method similar to Algorithm 1 to determine whether the current window can accommodate all the cells within it with the available widths of short rows and tall rows inside the window. We select the smallest window that can accommodate all the cells within it when cell version change of the flexible cells is allowed.

The reason why we want a hotspot to contain no more than three overflow bins is that the larger a hotspot, the larger is the final spreading window that we need and hence more cells will be relocated farther away from their positions calculated by the WL-lower-bound phase which will increase the final wirelength. As shown in Fig. 3(a), if all neighboring overflow bins is treated as a single hotspot, we need a 5x5 rectangular spreading window for it which may cause the cells to spread out far from the WL-lower-bound solution and cause unnecessary wirelength rise. Fig. 3 (b) shows that if we divide it into three hotspots, each can be resolved within a much smaller spreading window which helps to reduce the wirelength increase. Moreover, we do not want the aspect ratio of a window to be too large or too small. If we choose an inappropriate window, it may cause poor routability in the routing stage. We note that when we search for a suitable window for a hotspot, we allow the window to contain overflow bins belonging to some unprocessed hotspots.

C. Cell-to-Bin Allocation

After finding out the spreading window W of hotspot H, we try to spread the cells in W to eliminate local placement density overflow. We adopt a bipartitioning-based method. In each round, we divide W horizontally or vertically into two sub-windows and then allocate cells to these two sub-windows. We repeatedly divide the sub-window into two parts until each sub-window consists of one bin only.

When the vertical (horizontal) cutline is applied, we sort the cells according to their x-coordinates (y-coordinates) because we will do a bipartitioning according to the relative positions of the cells. Then, we want to find a suitable cutting point in the sorted cell list so that the cells in the two sub-lists will be assigned to the two sub-windows. We find the cutting point on the premise that the overflow degrees of the two sub-windows are the closest when we use the method described at the end of Section III-A to estimate the overflow degree of the two sub-windows. If the two sub-windows can have

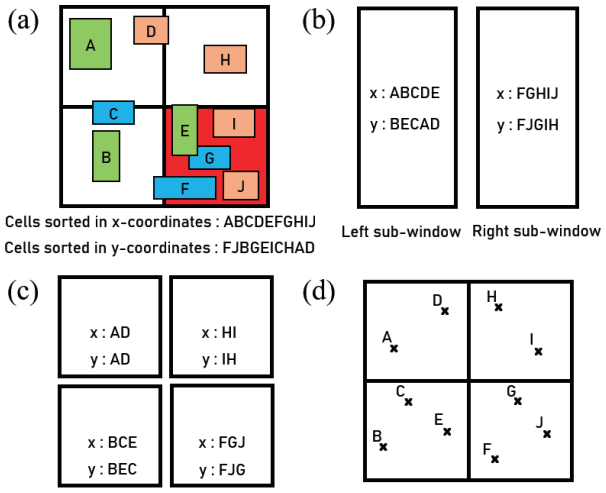

Fig. 4. An initial placement of 10 cells. (a) A hotspot consisting of the red bin only and its corresponding spreading window is the 2x2 window shown. (b) The result after vertical cut line is applied. The cell set is divided into the subset $\{A, B, C, D, E\}$ in the left sub-window and the subset $\{F, G, H, I, J\}$ in the right sub-window according to their order in the horizontal direction computed by the WL-lower-bound phase, and the utilization of the two sub-windows are close. (c) The final cell-to-bin allocation. The ordering of the cells in each bin according to their x/y-coordinates are computed. (d) The result of spreading anchors of cells within individual bins.

no overflow, we choose the cutting point that will make the usage rate of the two sub-windows the closest.

Fig. 4 gives an example. The blue cells are short-row cells including the SL-version and SH-version cells. The orange ones are tall-row cells, which are the TALL-version cells. The green cells are DR-version cells. Fig. 4(a) is an initial placement with ten cells, the hotspot consists of only one bin in red, and the spreading window W is composed of 2x2 bins. First, we divide window W into two sub-windows W_l and W_r when a vertical cut line is applied. Second, we sort the cells according to the x-coordinate and look for the cutting point for the sorted cell list $(A, B, C, D, E, F, G, H, I, J)$ and find that the best cutting point is between E and F. So, we assign $\{A, B, C, D, E\}$ to W_l and $\{F, G, H, I, J\}$ to W_r as shown in Fig. 4(b). Then we apply horizontal cut as shown in Fig. 4(c), so W_l is divided into sub-window W_{la} and sub-window W_{lb}, and W_r is divided into W_{ra} and W_{rb}. Next, we find the best cutting point for the cell list sorted in the y-coordinates within each of sub-windows W_l and W_r, so the cells are allocated to the smaller sub-windows accordingly.

D. Cell Anchor Spreading Within Bins

After allocating each cell to a desired bin, we still have to give each cell an anchor in its desired bin. We want to spread out the cell anchors in a bin evenly. Suppose $(c_1, ...c_n)$ is the list of cells sorted by their x-coordinates calculated in the WL-lower-bound phase, then we use the following formula to calculate the x-coordinates of all cell anchors in the bin:

$$x_i = b_x + \frac{\sum_{j=1}^{i-1} w_j}{\sum_{k=1}^{n} w_k} \times b_{width} \text{ , for } 1 \leqslant i \leqslant n \qquad (1)$$

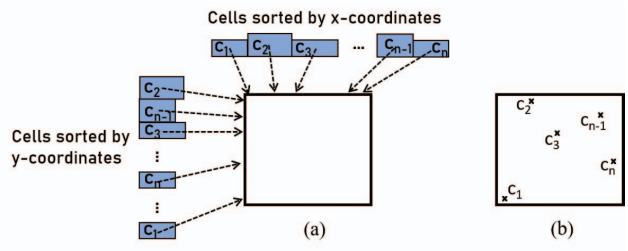

Fig. 5. (a) We want to spread out the cell anchors according to the relative orders of the cells in both horizontal and vertical directions. (b) The determined positions of the cell anchors.

where x_i the final x-coordinate of the anchor for cell c_i. w_i denotes the width of c_i. b_x denotes the x-coordinate of the bin's left boundary and b_{width} denotes the bin width. The final y-coordinate of the anchor for each cell can be computed based on their sorted y-coordinates calculated in the WL-lower-bound phase in a similar way. Note that the (x, y)-coordinates computed here will be used as the anchor of the corresponding cell for the next WL-lower-bound phase. Fig. 5 shows the idea of this scaling method. Fig. 4(d) shows the cell anchor positions after cell anchor spreading within individual bins in the previous example.

E. Update on Linear Systems

After finishing the WL-upper-bound phase, the cell anchors would be determined by the positions in Sec. III-D. Then we update all pseudo nets. In this step, we would set the weights of the pseudo nets according to [7] which helps accelerate convergence. Finally, for each net with 3 or more pins, the B2B net model is applied to update the set of its 2-pin nets based on the positions of cells in the WL-lower-bound phase at the last iteration. In the refined stage, the two linear systems would be updated with new anchors and new positions of cells at each iteration.

F. Cell Version Substitution of Flexible Cells

We use the overflow prediction of Sec. III-A to check the overflow situation of each bin. If a bin has no overflow in both tall-row and short-row resources based on the initial version of each cell inside it, we will not change the version of any cell in this bin, otherwise, we will change the versions of some flexible cells in it as follows. We compute the insufficient width of each resource in the bin based on the initial version of each cell inside it. We want to change the versions of some flexible cells so that there will be no overflow in either resource or the maximum of the insufficient widths of the two resources is minimized. For example, if the insufficient width of short-row resource is larger, we will first sort all the low-driving flexible cells in the bin whose initial versions are not TALL in descending order according to the distance between each of them to the center of the closest tall row. Then, we will change them one by one to TALL versions until there is no overflow in either resource or all of them have been changed or the insufficient widths of the two resources are nearly identical.

IV. Experimental Results

In this section, we first introduce how we generated the test cases for hybrid-row-height designs. Second, we describe the legalization method we used in our experiment. Finally, we show our experimental results.

A. Environment and Hybrid-Row-Height Benchmark Creation

We implemented our global placement approach, HybridGP, with C++ programming language. The experiments were conducted on a Linux machine with Ryzen 3.7GHz CPU and 256GB memory. We modified the benchmarks of ISPD2015 contest [2] for our experiments. Because cells in each benchmark of the ISPD2015 contest has the same height, we first modified the cell library so that each type of cell has four versions: SL, TALL, SH, and DR.

1) The SL version has the same height/width as the contest benchmarks.
2) The TALL version is 1.2 times the height of the SL version, and the width remains the same.
3) The width of the SH version is twice the width of the SL version and the height is the same as the SL version.
4) The height of the DR version is 2.2 times the height of the SL version (tall row height plus short row height), and the width is about 0.7 times the SL version.

We generated a set of benchmarks, where we set #SL:#TALL:#SH:#DR to 0.5:0.25:0.125:0.125, which means 50% of the cells are high-driving cells. Table I shows the information of the benchmarks we used, in which #cells, #nets, and #macros are the numbers of movable cells, pre-placed macros, and nets. For each benchmark, we randomly selected a subset of cells (about 20%-30%) to be the flexible cells whose versions can be changed. The row configuration of each benchmark was determined according to the short-row and tall-row resource requirements of the netlist so that there is no shortage in the placement region for both types of resources. We also scaled the sizes but kept the relative positions of macros and blockages in each design and other original features as much as possible.

B. Legalization Method

We need a legalizer to help evaluate the effectiveness of global placement. But since hybrid-row-height design is a new paradigm, no available legalizer exists. So we implemented a legalizer based on the multi-row global legalization (MGL) method in [6] while restricting that an SL-version or SH-version cell can only be placed in a short row, a TALL-version cell can only be placed in a tall row, and a DR-version cell can be placed on a row pair consisting of a short row and a tall row.

In MGL, each cell is legalized sequentially. First, for each cell c_i we consider a fixed window w_{c_i} centered at c_i. We start with a small window size. Next, we enumerate some legal positions in the corresponding window w_{c_i} for cell c_i and choose the one that minimizes the displacement of all cells located within w_{c_i}. If we cannot find a legal position for a cell, we will put it into the unplaced cell list and expand its

TABLE I
INFORMATION OF THE BENCHMARKS AND EXPERIMENTAL RESULTS. (G_WL AND L_WL DENOTE THE HALF-PERIMETER WIRELENGTH OF GLOBAL PLACEMENT AND LEGALIZATION RESULTS IN METERS, RESPECTIVELY. #UC DENOTES THE NUMBER OF UNPLACED CELLS. G_RT, L_RT, AND T_RT DENOTE THE GLOBAL PLACEMENT RUNTIME, LEGALIZATION RUNTIME, AND TOTAL RUNTIME IN SECONDS, RESPECTIVELY.)

Benchmark	#Cells	#Nets	#Macros	HybridGP					Baseline					
				G_WL	L_WL	G_RT	L_RT	T_RT	G_WL	L_WL	#UC	G_RT	L_RT	T_RT
mgc_des_perf_1	112644	112878	0	1.464	1.786	99	19	118	1.422	5.382	0	108	1825	1933
mgc_des_perf_a	108288	110281	4	1.724	1.988	88	18	107	1.693	3.045	0	101	199	300
mgc_des_perf_b	112644	112878	0	1.617	1.868	92	17	109	1.575	2.806	0	104	137	241
mgc_edit_dist_a	127413	131134	6	3.958	4.302	91	97	188	3.837	6.601	0	101	1849	1950
mgc_fft_1	32281	33307	0	0.474	0.590	20	5	25	0.464	1.232	0	22	201	223
mgc_fft_2	33281	33307	0	0.386	0.472	20	4	24	0.379	0.615	0	23	22	45
mgc_fft_a	30625	32088	6	0.624	0.709	21	5	26	0.611	1.096	0	24	83	107
mgc_fft_b	30625	32088	6	0.853	0.948	20	4	25	0.846	1.127	0	23	44	67
mgc_matrix_mult_1	155325	158527	0	2.408	2.936	118	29	147	2.356	9.530	0	123	7346	7469
mgc_matrix_mult_2	155325	158527	0	2.495	3.014	116	28	145	2.400	-	3047	126	-	-
mgc_matrix_mult_a	149650	154284	5	3.189	3.754	109	38	147	3.162	6.703	0	119	3000	3119
mgc_matrix_mult_b	146435	151612	7	2.911	3.329	108	27	135	2.849	6.056	0	120	2353	2473
mgc_matrix_mult_c	146435	151612	7	2.827	3.259	109	27	136	2.757	7.177	0	123	3888	4011
mgc_pci_bridge32_a	29517	29985	4	0.307	0.373	20	3	23	0.299	0.504	0	22	15	37
mgc_pci_bridge32_b	28914	29417	6	0.479	0.543	20	4	24	0.476	0.683	0	22	51	73
mgc_superblue11_a	925616	935613	1458	38.126	40.459	849	410	1259	38.196	-	22270	942	-	-
mgc_superblue12	1286948	1293413	89	32.490	35.008	1570	670	2240	32.091	41.950	0	1688	3925	5613
mgc_superblue14	612243	619697	340	25.062	26.797	533	589	1122	24.700	-	27712	582	-	-
mgc_superblue16_a	680450	697303	419	27.106	28.832	563	236	799	26.864	33.633	0	635	5720	6355
mgc_superblue19	506097	511606	286	16.189	17.150	443	174	618	15.949	21.121	0	484	4123	4607
normalized				1.019	0.644	0.90	0.07	0.23	1.0	1.0		1.0	1.0	1.0

window in the next round to find a legal and suitable position. The above steps are repeated until the window size reaches a user defined limit, and we will report the remaining number of unplaced cells if we cannot get a legal placement solution.

C. Results

Our experimental flow is as follows. We compare (i) the result of running HybridGP followed by the legalizer described above, and (ii) the result of the baseline method followed by the same legalizer. The baseline method is a modified version of HybridGP, and behaves like a conventional global placer. The baseline method compares the available area and the total area of cells placed within a bin to determining whether a bin is overflow or not. In addition, it does not execute the cell version substitution step.

Table I shows the results for the wirelength of global placement (G_WL), the wirelength of legalization (L_WL), the number of unplaced cells (#UC), the runtime of global placement (G_RT), the runtime of legalization (L_RT), and the total runtime (T_RT). Because the baseline method did not take into account the local usage of tall and short row resources, it required a lot of effort and a large amount of cell movement to legalize which was detrimental to the wirelength. The flow with the baseline method also has unplaced cells after legalization for three benchmarks, but it did not happen to the flow with HybridGP. Our flow has an improvement of 35.6% in the final legalized wirelength compared to the flow using the baseline method. Besides, the total runtime of our flow is 77% less than the flow using the baseline method. Note that we only counted the testcases without unplaced cells in computing the average improvement.

V. CONCLUSIONS

In this paper, we present a global placer that can handle hybrid-row-height designs. It considers the usage of short-row and tall-row resources and changes cell versions to facilitate the subsequent legalization stage. The experimental results well support our placer. As far as we know, it is the first global placer in the literature to focus on advanced hybrid-row-height designs.

REFERENCES

[1] S.-H. Baek, H.-Y. Kim, Y.-K. Lee, D.-Y. Jin, S.-C. Park, and J.-D. Cho. Ultrahigh density standard cell library using multi-height cell structure. In *Proc. of SPIE - The International Society for Optical Engineering*, volume 7268, 12 2008.

[2] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis. Ispd 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *Proc. of International Symposium on Physical Design*, page 157–164, 2015.

[3] J. Chen, Z. Huang, Y. Huang, W. Zhu, J. Yu, and Y.-W. Chang. An efficient epist algorithm for global placement with non-integer multiple-height cells. In *Proc. of Design Automation Conference*, 2020.

[4] S. Dobre, A. B. Kahng, and J. Li. Mixed cell-height implementation for improved design quality in advanced nodes. In *Proc. of International Conference on Computer-Aided Design*, pages 854–860, 2015.

[5] Andrew B. Kahng. Advancing placement. In *Proc. of International Symposium on Physical Design*, page 15–22, 2021.

[6] H. Li, W.-K. Chow, G. Chen, E. F. Y. Young, and B. Yu. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In *Proc. of Design Automation Conference*, 2018.

[7] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev. Polar: A high performance mixed-size wirelengh-driven placer with density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(3):447–459, 2015.

[8] Y.-H. Lin, C.-H. Wang, and Y.-T. Hou. Variant cell height integrated circuit design. *U.S. Patent 10 878 157*, Dec. 2020.

[9] P. Spindler, U. Schlichtmann, and F. M. Johannes. Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1398–1411, 2008.

[10] N. Viswanathan and C.C.-N. Chu. Fastplace: efficient analytical placement using cell shifting, iterative local refinement,and a hybrid net model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(5):722–733, 2005.

[11] Y.-C. Zhao, Y.-C. Lin, T.-C. Wang, T.-H. Wang, Y.-R. Wu, H.-C. Lin, and S.-Y. Kao. A mixed-height standard cell placement flow for digital circuit blocks. In *Proc. of Design, Automation and Test in Europe Conference Exhibition*, pages 328–331, 2019.