Floorplan Management: Incremental Placement for Gate Sizing and Buffer Insertion *

Chen Li, Cheng-Kok Koh
School of ECE, Purdue University
West Lafayette, IN 47907, USA
{li35, chengkok}@ecn.purdue.edu

Patrick H. Madden
CS Department, Binghamton University
Binghamton, NY 13902, USA
pmadden@cs.binghamton.edu

ABSTRACT

Incremental physical design is an important methodology towards achieving design closure for high-performance large-scale circuits. Placement tools must accommodate incremental changes to the layout and netlist due to physical synthesis techniques without perturbing the original metrics. We present an incremental placement approach using floorplan sizing to manage the resources and demands of the whole chip region in order to accommodate the changes due to gate sizing and buffer insertion. The experimental results show that this approach can accommodate a wide range of incremental changes without a loss in wirelength and routability. Most important, it also maintains the stability of a placement such that the convergence of physical synthesis iterations can be greatly enhanced.

1. INTRODUCTION

The design of high-performance circuits typically requires iterations of the design flow to correct mistakes made earlier or accommodate changes made later in the flow. The correction of timing violations discovered late in the physical design steps for example, may require iterations of the design flow. In addition, placement algorithms may generate unroutable placements due to routing congestions in some regions. Moreover, buffers inserted to optimize the timing may increase the layout area greatly [12]. The unroutability of a placement and the change of chip area also necessitate iterations of the design flow. To achieve design closure of high-performance circuits, physical synthesis, which combines logical and physical optimizations [4], has been proposed and widely used in current design flow. There is a growing trend to perform physical synthesis during the placement flow, as interconnect delay can be estimated more accurately and physical synthesis techniques can adapt to the changes of interconnects. In [6], simultaneous gate sizing and placement are performed for improving the timing of critical paths. In [13], the netlist is transformed during the placement process. In [11], buffers are modeled explicitly as repulsive forces and attractive forces during analytical placement process to ensure the convergence of placement at 45nm and 32nm technology nodes.

These optimizations due to physical synthesis inevitably change the layout or netlist, thereby making the placements illegal. Consequently, local white space is required to accommodate these changes. For example, a huge number of buffers may require white space locally everywhere over the whole chip region. However, local resources are not always enough. We must either reserve enough local white space to accommodate these incremental changes, or manage layout resources of the whole chip region effectively such that white space can migrate from resource-abundant regions to resource-hungry regions. The white space reservation approach typically suffers from a larger layout area and longer interconnects due to the presence of unnecessary white space. Worse still, predefined or uniformly distributed buffer blocks or white space before the placement stage may not be good candidate locations for these buffers in the placement [12]. As another run of placement flow may invalidate the effectiveness of the physical synthesis, the white space migration approach relies on incremental physical design methodology [7]. In this work, our focus is on the incremental placement techniques to facilitate changes made to the layout and netlist.

The challenge of incremental placement is that it should be stable; a placement solution after incremental changes should be similar to the original placement with minimal perturbation so as to preserve the high quality of the original placement and maintain the physical information that physical synthesis uses for optimization. Stability of a placement is quantified by maximum and average of cell displacements in [1]. In [1], in order to prevent from generating totally different placements from run to run, some cells are chosen to be tethered within some regions to obtain stable placements. However, this method still perturbs the placement greatly and might not be effective in accommodating incremental changes.

Another issue related to incremental placement is whether the core region changes during the placement flow. Fixed-die placement assumes the the core region reserves enough white space to accommodate changes due to physical synthesis, whereas variable-die placement changes the core region accordingly to avoid wasteful white space or preserve some metrics [14].

In [10], a white space allocation approach is proposed to allocate white space to congested regions so as to improve routability of placements. In this paper, we extend this approach and propose a floorplan management approach to re-allocate the layout resources to match the demands due to gate sizing, buffer insertion, and routing congestion. As gate sizing and buffer insertion are likely to change the area requirement, we consider variable-die placement. The main contribution of our work is that we propose a stable incremental placement approach that can accommodate incremental changes to the layout and netlist while maintaining the good quality of the original placement solution. Experimental results show that compared to a commercial placement tool (CPT), our stable incremental approach can handle a wide range of incremental changes due to gate sizing with 23% shorter routed wirelength and better routability. Compared to another run of CPT or Dragon on buffered netlist, our approach can generate placements without buffer violation (defined in Sec. 4) and with shorter routed wirelengths and comparable routability. Most important, stability of placements can

*This research is partially supported by National Science Foundation under Award CCR-9984553, Semiconductor Research Corporation under Project 947.1, and an IBM Faculty Partnership Award.

be maintained; our approach perturbs the net lengths within 1% of core region bounding box on average.

2. FLOORPLAN MANAGEMENT

We illustrate the basic idea underlying our approach through Figure 1. A set of cells (depicted as rectangles) contained in this region and local interconnects or bypass interconnects (connecting the cells) are the demands on this region. The available area of this region and available routing tracks on the routing layers above this region are the resources of this region. Physical synthesis techniques such as gate sizing, buffer insertion or other local netlist changes may result in changes in the demands on this region. For the case of increasing demands, we can either move some cells out of this region to reduce the demands or increase the resources of the region by expanding the region. As the number of routing tracks is typically proportional to the area of the region, we can simply enlarge the region to accommodate the changes due to the increase in both area and routing demands. We propose a simple, yet effective approach to adjust the allocation of all resources such that each region has enough resources to meet its demands.

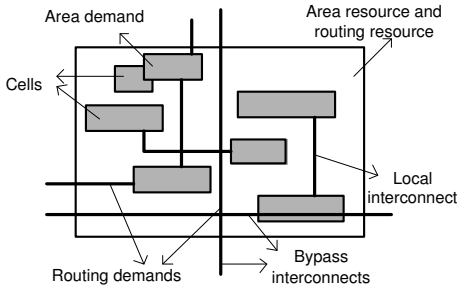


Figure 1: Floorplan management of a single region.

We represent the placement as a floorplan by a slicing tree. The main advantages are: (i) This floorplan captures the relative positions of cells; thus, it maintains the quality of the existing placement. (ii) This approach is able to adjust the resources of the whole layout even when nearby resources are not enough for local changes. While most other incremental placement approaches may use only the local available resources to reduce the perturbation.

We assume the following flow in this work: Starting from a valid placement, physical synthesis techniques are performed without consideration of the legality of the placement. For example, cells are sized on site and buffers are inserted with overlaps with other cells. The core region is scaled accordingly such that there is enough area to place all the cells. Thus, we perform a variable-die incremental placement. In this work, our focus is not on improving the existing placements by physical synthesis techniques. Instead, we focus on incremental placement approaches to accommodate changes due to physical synthesis techniques.

Our approach contains the following steps.

(1) Represent the placement after physical synthesis as a floorplan by a slicing tree.

(2) Analyze the area demands of every floorplan region.

(3) Size the floorplan regions to match the estimated demands.

(4) Apply detailed placer to legalize and further optimize the placement.

2.1 Floorplan representation

We represent the placement after physical synthesis as a floorplan by a slicing tree. We recursively cut the placement until every region contains only a small number of cells. Given an initial placement, cut directions are determined by the aspect ratio of this region. Cut locations are at horizontal or vertical center of the region. For a region, once cut direction and location are determined, all cells located at the left (if cut vertically) or at the above (if cut horizontally) of the cut line form left child of that tree node. The rest cells in the region form right child of that node.

Area demands of each floorplan region is the total area of the cells contained in this region, which include inserted buffers and cell size changes due to gate sizing. Congestion demands can be estimated and converted into area demands (see Sec. 2.2) for these floorplan regions.

2.2 Floorplan sizing

After performing demand analysis on all tree nodes, we size all floorplan regions in a top-down fashion along the slicing tree such that the layout area allocated to each region matches its demands. We illustrate this flow as in Figure 2 (b). Consider a region R with lower-left corner (x_0, y_0), upper-right corner (x_1, y_1) and the original vertical cut direction at $x_{cut} = (x_0 + x_1)/2$. Thus, the area of this region is $A_R = (x_1 - x_0)(y_1 - y_0)$. Assume that the total area of cells after physical synthesis for left subregion R_0 and right subregion R_1 are S_0 and S_1. Note that $(S_0 + S_1) \leqslant A_R$ as the core region has been expended. We adjust the cut line such that the areas of two subregions can accommodate the cells contained in them. If areas of subregions allocated are proportional to their area demands, the new cut line location x'_{cut} can be obtained as follows:

$$\gamma = \frac{S_0}{S_0+S_1}, \tag{1}$$

$$x'_{cut} = \gamma x_1 + (1-\gamma)x_0, \tag{2}$$

where γ is the ratio of the left subregion area to A_R after the cut line adjustment.

Such an approach allows us to handle any demand that can be met by area sizing. For example, we can perform congestion-driven white space allocation simultaneously as in [10]. Assume the congestion levels for two subregions are CG_0 and CG_1 respectively. We want to distribute the white space, which is $(A_R - S_0 - S_1)$, to the two subregions, such that the amounts of white space in the two subregions are proportional their congestion levels. Thus, the amount of white space allocated to subregion R_0 is $(A_R - S_0 - S_1)\frac{CG_0}{CG_0+CG_1}$.

$$\gamma = \frac{S_0 + (A_R - S_0 - S_1)\frac{CG_0}{CG_0+CG_1}}{A_R}. \tag{3}$$

2.3 Detailed placement

We apply a detailed placer that can preserve white space distribution and further reduce HPWL (half-perimeter wirelength). First, a greedy legalization algorithm [8] is applied to remove the overlaps, Then, a sliding window based local minimization approach [2] is applied to minimize HPWL. We slide a single-row or double-row window across the entire chip, and perform cell swappings within the window to optimize the wirelength.

3. APPLICATION TO GATE SIZING

We apply our floorplan management approach to perform incremental placement for gate sizing. We consider two different problems. (i) Given an initial placement in which a certain number of cells are re-sized, find a legal placement that maintains the good metrics of the original placement. This is a typical incremental

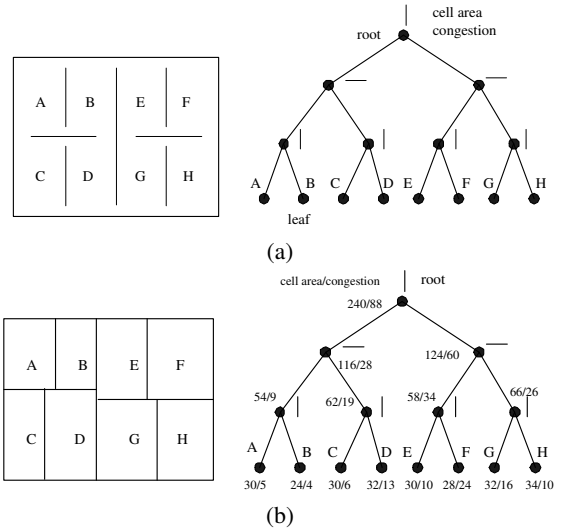


Figure 2: (a) A floorplan representation. (b) Floorplan sizing after demand analysis. Cut lines are adjusted top-down in the slicing tree according to the area demands. .

placement problem for gate sizing during the placement stage. (ii) Assume a placement of a circuit with uniform sized cells is given. Now given the non-uniform width of every cell, find a legal placement that maintains the good metrics of the original placement. This is a special case of gate sizing in which every cell changes its width. Such uniform sized placement benchmarks are used in various placement research such as PEKO [5] benchmarks and some versions of IBM benchmarks [3]. Some placers, mPL [3] for example, is very effective in placing uniform sized cells. The prototype placer Warp1 [15] may also consider uniform sized cells. Essentially, this is a radically different approach for placement process: We first transform the original netlist to contain uniform sized cells. Then, we obtain a placement for this uniform sized circuit. Finally, we transform the placement to accommodate the original netlist.

3.1 Gate sizing

We use IBM-Place v2 easy benchmarks [16] in this experiment. We randomly choose a subset of cells in each benchmark and upsize to double their widths. At the same time, we enlarge the chip dimension horizontally and vertically to maintain the aspect ratio and the relative amount of white space. We evaluate our approach for various numbers of up-sized cells (1%, 5%, 10%, 30%, 50% of the total cell count). For clarity, we call the original IBM-Place v2 benchmarks IBM benchmarks, and these up-sized benchmarks IBM-1%, IBM-5%, and so on. The initial placements are obtained by one run of Dragon 3.01 [16] on the IBM benchmarks, as Dragon can produce placements with good routability and routed wirelengths.

The experimental results are shown in Table 1. In this table, column 'IBM' shows the routing results of Dragon's placements of original IBM benchmarks. The routing results include routed wirelength ('r-WL'), number of violations ('vlts') and over-capacity gcells ('cg%') reported by Cadence WROUTE. Note that the wirelengths are scaled by 10^6 micron for all tables. Routing status can be successful (with no violations), finished (with smaller number of violations), and failed (with too many violations or too much routing time). Columns 'IBM-x%' also show the routing results of placements incrementally generated by our approach when x% of cells are up-sized to their double widths. We are particularly interested in the amount of net length changes as a ratio of the change in half-perimeter wirelength (HPWL) to the core region bounding box after gate sizing. We report the average ('ave'), maximum ('max'), and standard deviation ('s.d.') of net length changes in terms of percentages of core region bounding box. Note that Adya et al. [1] use maximum and average cell displacements to evaluate the stability of a placement tool. We argue that net length changes are a more accurate metric for the stability study of the placement flow as they reflect the changes in delay and routability. We do not show the statistics of cell displacements in this table due to space limitation. In the 'summary' row, we show the average ratios among routed wirelengths with respect to those on IBM-1% and average of the statistics of net length changes over all benchmarks.

Table 1 show that our approach can effectively handle various numbers of up-sized cells. Routed wirelengths on IBM-1% are smaller than Dragon's results on original IBM benchmarks, because our floorplan management approach can further reduce the wirelength. The routed wirelengths gradually increase as the number of up-sized cells due to the expanded cells and core region, but at a rate slower than core region expansion rate. Simultaneously, we preserve the low congestion level (which can be seen from over-capacity gcells 'cg%'), thus achieving comparable routability. Most routings are successful except for only one or two finished routings on IBM-5%, IBM-10%, and IBM-50%, whereas Dragon has one failed routing on the original IBM benchmarks. The average net length changes are low, which are smaller than 1% of core region bounding box; we maintain the stability of the placements.

3.2 Placement transformation

We construct benchmarks with uniform sized cells from IBM-Place v2 easy benchmarks (IBM) [16] by using the average width of all cells for every cell. We call these uniform sized benchmarks U-IBM benchmarks. U-IBM benchmarks maintain the same core area as the original IBM benchmarks. For evaluating various placement tools on uniform sized benchmarks and the effectiveness of our floorplan management approach, we run a few latest academic placers, including Dragon 3.01 [16], CAPO 8.8 [2], mPL-R [10], and a commercial placement tool (CPT) to obtain the initial placements on these U-IBM benchmarks. Dragon and CAPO are run five times and mPL-R and CPT once.

Table 2 shows the routing results of Dragon's placements of IBM benchmarks and our gate sizing results starting from various initial placements. All data are obtained by averaging over the runs except column 'S/V/F'. In this table, we compare the stability of our approach by evaluating the changes in both cell locations and net lengths. We report the average, maximum, and standard deviation of cell displacements and net length changes in terms of percentages of the core region bounding box. Column 'S/V/F' shows the number of successful routings (denoted as 'S'), finished routings (denoted as 'V'), and failed routings (denoted as 'F') on five runs (for Dragon and CAPO) or one run (for mPL-R and CPT) of each placement flow. In the 'summary' row, we show the average of 'HPWL', 'r-WL', 'cg%' normalized to those of Dragon on IBM benchmarks, average 'vlts', and average stability metrics over all benchmarks. 'S/V/F' on this row show the total 'S/V/F' for each flow. For comparison, we also show the routing results of CAPO, mPL-R, and CPT on original IBM benchmarks, but only in the 'summary' row.

On the average, the average and maximum cell displacement (net length changes) are 2.1% and 9.3% (0.5% and 8.9%) of the core region bounding box over all tools on all benchmarks. Compared to the "tether" approach proposed in [1], in which average

Table 1: Results of gate sizing. Initial placements are obtained by Dragon on IBM benchmarks.

bench	IBM						IBM-1%						IBM-5%					
	net length changes			routing results			net length changes			routing results			net length changes			routing results		
-marks	ave	max	s.d.	r-WL	vlts	cg%	ave	max	s.d.	r-WL	vlts	cg%	ave	max	s.d.	r-WL	vlts	cg%
ibm01e	-	-	-	0.930	6234	5.93	0.6	6.5	0.7	0.779	0	1.98	0.6	8.2	0.7	0.791	0	1.57
ibm02e	-	-	-	2.25	0	0.88	0.5	18.5	0.6	2.07	0	0.69	0.5	33.8	0.8	2.08	0	0.48
ibm07e	-	-	-	4.55	0	0.70	0.4	9.0	0.4	4.13	0	0.25	0.4	5.6	0.4	4.18	0	0.16
ibm08e	-	-	-	4.67	0	0.06	0.3	8.7	0.4	4.37	0	0.04	0.3	6.7	0.4	4.40	0	0.03
ibm09e	-	-	-	3.80	0	0.02	0.3	4.5	0.4	3.53	0	0.01	0.3	6.1	0.4	3.58	1	0.02
ibm10e	-	-	-	7.53	0	0.05	0.3	4.9	0.4	6.94	0	0.02	0.3	6.1	0.4	7.03	0	0.01
ibm11e	-	-	-	5.72	0	0.04	0.3	4.2	0.4	5.32	0	0.04	0.3	7.6	0.3	5.39	0	0.03
ibm12e	-	-	-	10.4	0	0.20	0.4	5.6	0.4	9.62	0	0.08	0.3	5.0	0.4	9.74	0	0.05
summary				1.096			0.4	7.7	0.5	1.000			0.4	9.9	0.5	1.012		

bench	IBM-10%						IBM-30%						IBM-50%					
	net length changes			routing results			net length changes			routing results			net length changes			routing results		
-marks	ave	max	s.d.	r-WL	vlts	cg%	ave	max	s.d.	r-WL	vlts	cg%	ave	max	s.d.	r-WL	vlts	cg%
ibm01e	0.6	25.2	0.8	0.800	0	1.38	0.7	29.0	1.1	0.827	0	0.58	0.8	20.8	1.1	0.847	0	0.19
ibm02e	0.5	39.8	0.7	2.09	0	0.24	0.8	22.3	1.1	2.19	0	0.04	1.0	19.8	1.4	2.29	1	0.01
ibm07e	0.4	8.1	0.4	4.23	0	0.09	0.5	18.1	0.8	4.51	0	0.01	0.7	20.7	1.2	4.80	0	0.00
ibm08e	0.3	6.8	0.4	4.49	0	0.02	0.5	9.5	0.7	4.77	0	0.01	0.6	13.3	1.1	5.09	0	0.00
ibm09e	0.3	4.3	0.4	3.67	0	0.01	0.4	8.3	0.6	3.96	0	0.01	0.5	11.6	0.9	4.25	0	0.00
ibm10e	0.3	6.1	0.4	7.27	1	0.01	0.4	8.4	0.6	7.70	0	0.00	0.6	11.6	0.9	8.23	1	0.00
ibm11e	0.3	5.1	0.4	5.48	0	0.02	0.4	9.1	0.6	5.88	0	0.01	0.5	12.9	0.9	6.32	0	0.00
ibm12e	0.4	6.0	0.4	9.90	0	0.02	0.5	8.2	0.7	10.6	0	0.00	0.7	13.0	1.0	11.3	0	0.00
summary	0.4	12.7	0.5	1.028			0.5	14.1	0.8	1.093			0.7	15.5	1.0	1.159		

cell displacements ranged from 2.6% to 5.1% and maximum cell displacements ranged from 43% to 58% were reported, we are able to maintain a better stability during our floorplan management approach, even though we simultaneously consider layout changes due to congestion reduction and gate sizing. In contrast, the results in [1] are obtained from different placement runs of the same netlists. These experimental results show that our approach can effectively handle the overlaps resulting from changes of cell widths.

Figure 3 shows the distribution of cell displacements and net length changes of our approach. In this figure, initial placement is obtained by one of Dragon runs on uniform ibm02-easy benchmarks. Net length changes are mostly smaller than 2% of the core region bounding box.

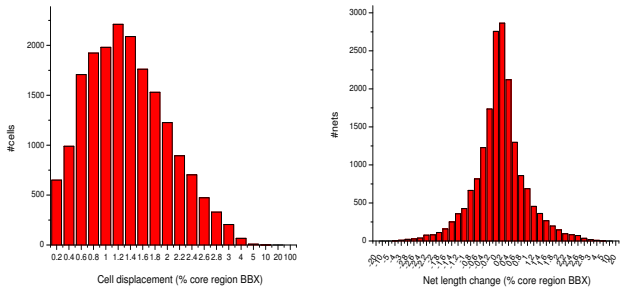
From the summary row in Table 2, we observe that our routed wirelengths are smaller than those obtained by Dragon, CAPO, mPL-R, and CPT on original IBM benchmarks by 1.7% to 7.7% while preserving or improving the routability. By transforming uniform placements to non-uniform placements, we are able to obtain placements with comparable or better routed wirelength and routability. These results also show that such a radically different placement approach by transforming uniform sized placements to non-uniform sized placement might be a feasible approach as handling circuits with uniform sized cells might simplify the placement tools. Our incremental placement approach is effective in facilitating placement transformation in terms of routed wirelength, routability and stability.

For comparison, we also run CPT in ECO mode to legalize the placements for this transformation. Initial placements are obtained by running CPT on the U-IBM benchmarks. We compare our floorplan management approach with CPT in ECO mode in table 3. Runtimes of our approach and CPT in ECO mode are also shown in the column 'runtime'. In the 'summary' row, we also show HPWL normalized to that of the initial placement and the average runtime. It shows that CPT cannot effectively handle the huge number of overlaps resulted from this problem. Our floorplan management approach obtains 23% better routed wirelength and significantly better routability. The runtime of our approach is comparable to CPT in ECO mode.

4. APPLICATION TO BUFFER INSERTION

Table 3: Comparison of our approach and CPT in ECO mode on placement transformation. Initial placements are obtained by CPT on U-IBM benchmarks.

bench-marks	Initial HPWL	Approach	HPWL	run-time	Cadence WROUTE report			
					S/V/F	r-WL	vlts	cg%
ibm01e	0.580	our approach	0.579	40s	0/1/0	0.782	1	1.64
		CPT ECO	0.787	16s	0/0/1	1.266	65323	15.78
ibm02e	1.55	our approach	1.56	90s	1/0/0	2.15	0	0.94
		CPT ECO	1.69	15s	0/0/1	2.48	12197	3.95
ibm07e	3.62	our approach	3.63	145s	1/0/0	4.38	0	0.18
		CPT ECO	5.70	400s	0/0/1	7.30	786696	32.17
ibm08e	3.98	our approach	3.96	233s	0/1/0	5.14	1	0.37
		CPT ECO	4.68	98s	0/0/1	6.69	103192	12.77
ibm09e	3.39	our approach	3.38	171s	1/0/0	3.88	0	0.02
		CPT ECO	4.35	113s	0/0/1	5.54	1719	3.21
ibm10e	6.49	our approach	6.56	267s	0/1/0	7.79	1	0.07
		CPT ECO	7.66	358s	0/0/1	10.64	119504	9.73
ibm11e	5.06	our approach	5.10	213s	1/0/0	5.70	0	0.03
		CPT ECO	5.84	121s	0/1/0	7.01	0	0.61
ibm12e	9.27	our approach	9.34	332s	1/0/0	11.4	0	0.40
		CPT ECO	9.77	75s	0/1/0	12.34	2	1.71
summary	1.000	our approach	1.004	186s	5/3/0	1.000	3	1.000
		CPT ECO	1.234	150s	1/2/5	1.356	1088633	68.90



(a) (b)

Figure 3: Stability results of our approach on ibm02e benchmark. (a) cell displacement distribution before and after our approach. (b) net length change distribution before and after our approach.

Table 2: Results of placement transformation. Initial placements are obtained by Dragon, CAPO, mPL-R, and CPT on U-IBM benchmarks.

bench	Initial Placement		Our Approach	cell displacements			net length changes			Cadence WROUTE report			
-marks	Placer	HPWL	HPWL	ave	max	s.d.	ave	max	s.d.	S/V/F	r-WL	vlts	cg%
ibm01e	Dragon on IBM	0.594	-	-	-	-	-	-	-	1/0/4	0.929	5346	5.64
	Dragon on U-IBM	0.630	0.571	2.6	21.9	1.6	0.9	20.3	1.3	2/3/0	0.820	8.6	2.72
	CAPO on U-IBM	0.552	0.569	2.2	8.4	1.2	0.7	8.7	0.8	4/1/0	0.797	0.2	2.70
	mPL-R on U-IBM	0.500	0.553	4.7	12.8	2.6	0.7	10.3	1.1	1/0/0	0.761	0	1.43
	CPT on U-IBM	0.580	0.579	3.0	8.2	1.4	0.6	6.7	0.7	0/1/0	0.782	1	1.64
ibm02e	Dragon on IBM	1.57	-	-	-	-	-	-	-	5/0/0	2.18	0	1.07
	Dragon on U-IBM	1.51	1.51	1.4	18.6	0.8	0.6	18.5	0.6	5/0/0	2.04	0	0.89
	CAPO on U-IBM	1.57	1.60	1.7	7.4	0.9	0.7	6.7	0.7	4/1/0	2.23	3.8	1.30
	mPL-R on U-IBM	1.44	1.49	1.6	5.7	0.8	0.6	6.4	0.7	1/0/0	1.98	0	0.44
	CPT on U-IBM	1.55	1.56	1.9	7.0	0.9	0.6	8.5	0.7	1/0/0	2.15	0	0.94
ibm07e	Dragon on IBM	3.59	-	-	-	-	-	-	-	4/1/0	4.55	4.4	0.68
	Dragon on U-IBM	3.66	3.57	1.4	8.1	0.8	0.5	7.7	0.6	4/1/0	4.23	0.2	0.23
	CAPO on U-IBM	3.60	3.80	1.9	16.2	1.0	0.5	16.0	0.8	5/0/0	4.59	0	0.46
	mPL-R on U-IBM	3.02	3.39	3.8	11.0	2.3	0.6	11.2	1.0	1/0/0	3.94	0	0.14
	CPT on U-IBM	3.62	3.63	1.8	7.5	0.8	0.4	6.6	0.5	1/0/0	4.38	0	0.18
ibm08e	Dragon on IBM	3.69	-	-	-	-	-	-	-	5/0/0	4.78	0	0.08
	Dragon on U-IBM	3.65	3.61	1.2	7.8	0.6	0.4	9.1	0.4	4/1/0	4.49	0.2	0.06
	CAPO on U-IBM	3.82	3.92	1.5	7.9	0.8	0.4	8.0	0.5	2/3/0	4.87	0.6	0.16
	mPL-R on U-IBM	3.47	3.73	2.9	8.8	1.8	0.4	8.0	0.7	1/0/0	4.72	0	0.18
	CPT on U-IBM	3.98	3.96	1.7	10.6	0.9	0.4	9.0	0.5	0/1/0	5.14	1	0.37
ibm09e	Dragon on IBM	3.25	-	-	-	-	-	-	-	5/0/0	3.81	0	0.03
	Dragon on U-IBM	3.23	3.13	1.6	10.1	0.8	0.4	10.0	0.5	5/0/0	3.51	0	0.02
	CAPO on U-IBM	3.23	3.35	1.6	6.1	0.8	0.4	6.8	0.6	5/0/0	3.72	0	0.02
	mPL-R on U-IBM	2.95	3.22	3.5	9.1	1.5	0.5	10.3	0.7	1/0/0	3.61	0	0.02
	CPT on U-IBM	3.39	3.38	2.0	7.9	1.1	0.4	6.8	0.5	1/0/0	3.88	0	0.02
ibm10e	Dragon on IBM	6.28	-	-	-	-	-	-	-	3/2/0	7.46	7.2	0.04
	Dragon on U-IBM	6.00	5.96	1.5	5.5	0.8	0.4	5.5	0.4	3/2/0	6.75	0.4	0.01
	CAPO on U-IBM	6.26	6.36	1.5	7.9	0.9	0.4	7.4	0.5	3/2/0	7.22	0.4	0.04
	mPL-R on U-IBM	5.81	6.34	2.5	7.4	1.4	0.5	8.0	0.7	1/0/0	7.30	0	0.10
	CPT on U-IBM	6.49	6.56	1.4	9.3	0.8	0.4	8.0	0.5	0/1/0	7.79	1	0.07
ibm11e	Dragon on IBM	4.94	-	-	-	-	-	-	-	2/3/0	5.68	0.8	0.05
	Dragon on U-IBM	4.81	4.72	1.2	6.7	0.7	0.4	6.7	0.4	5/0/0	5.16	0	0.02
	CAPO on U-IBM	4.76	4.91	1.5	7.4	0.8	0.4	7.2	0.5	4/1/0	5.39	0.2	0.04
	mPL-R on U-IBM	4.50	4.89	2.3	7.3	1.5	0.4	7.3	0.6	0/1/0	5.42	1	0.08
	CPT on U-IBM	5.06	5.10	1.0	5.5	0.6	0.4	6.5	0.4	1/0/0	5.70	0	0.03
ibm12e	Dragon on IBM	8.69	-	-	-	-	-	-	-	3/2/0	10.6	0.4	0.22
	Dragon on U-IBM	8.91	8.60	1.5	9.5	0.8	0.5	10.0	0.6	4/1/0	10.2	0.2	0.18
	CAPO on U-IBM	8.47	8.85	1.8	10.7	1.1	0.5	8.4	0.7	2/3/0	10.4	19	0.21
	mPL-R on U-IBM	7.13	8.10	4.8	13.5	3.4	0.6	8.5	0.9	1/0/0	9.51	3	0.07
	CPT on U-IBM	9.27	9.34	1.1	5.2	0.7	0.4	5.3	0.4	1/0/0	11.4	0	0.40
summary	Dragon on IBM	1.000	-	-	-	-	-	-	-	28/8/4	1.000	670	1.00
	CAPO on IBM	0.999	-	-	-	-	-	-	-	15/14/11	1.016	2555	4.28
	mPL-R on IBM	0.967	-	-	-	-	-	-	-	8/0/0	0.937	0	0.75
	CPT on IBM	1.029	-	-	-	-	-	-	-	6/2/0	1.035	15.5	2.44
	Dragon on U-IBM	-	0.969	1.5	11.0	0.9	0.5	11.0	0.6	32/8/0	0.923	1.2	0.60
	CAPO on U-IBM	-	1.019	1.7	9.0	0.9	0.5	8.7	0.6	29/11/0	0.972	3.0	0.95
	mPL-R on U-IBM	-	0.969	3.3	9.4	1.9	0.5	8.8	0.9	6/2/0	0.920	0.5	1.00
	CPT on U-IBM	-	1.030	1.7	7.7	0.9	0.4	7.2	0.5	5/3/0	1.001	0.4	1.38

Table 4: Results of buffer insertion. Initial placements are obtained by Dragon on IBM benchmarks.

bench	Initial	After buffer			Our approach										CPT on buffered netlist					Dragon on buffered netlist				
	placement	insertion				cell displacements			net length changes			routing results			run		routing results			run		routing results			run
-marks	bv	bv	bufs	A	bv	ave	max	s.d.	ave	max	s.d.	r-WL	vlts	cg%	time	bv	r-WL	vlts	cg%	time	bv	r-WL	vlts	cg%	time
ibm01e	201	0	0.038	1.016	0	1.5	20.2	0.9	0.6	19.4	0.9	0.783	0	3.11	85s	1	0.883	0	3.54	160s	8	0.906	0	4.98	637s
ibm02e	997	0	0.165	1.078	0	0.9	17.6	0.6	0.5	5.5	0.5	2.24	4	2.74	177s	43	2.24	0	2.55	346s	37	2.33	0	1.17	1292s
ibm07e	2676	0	0.185	1.089	0	1.5	5.5	0.8	0.4	4.5	0.4	4.39	2	1.94	319s	978	5.00	0	0.25	723s	195	4.17	0	0.09	3291s
ibm08e	2607	0	0.203	1.101	0	0.7	6.3	0.5	0.3	4.6	0.3	4.65	1	0.20	461s	1021	5.46	1	0.08	922s	490	4.92	0	0.06	3751s
ibm09e	1991	0	0.114	1.056	0	0.7	7.5	0.4	0.3	3.8	0.4	3.63	0	0.03	352s	408	4.02	0	0.02	858s	390	3.80	0	0.02	3041s
ibm10e	4830	0	0.250	1.094	0	1.1	4.8	0.5	0.3	4.8	0.3	7.35	2	0.09	562s	1344	7.93	0	0.01	1274s	885	7.68	0	0.03	4449s
ibm11e	3228	0	0.167	1.084	0	0.7	5.0	0.4	0.3	3.3	0.3	5.55	0	0.11	450s	930	5.96	0	0.02	1098s	489	5.67	0	0.04	3341s
ibm12e	7241	0	0.404	1.141	0	0.9	5.4	0.6	0.3	3.7	0.4	10.5	0	0.30	727s	3172	11.9	2	0.23	1445s	1721	11.5	0	0.29	8733s

In this experiment, we evaluate the effectiveness of our floorplan management approach to accommodate changes due to inserted buffers. Given a initial placement in which some nets are changed due to buffer insertion, find a legal placement that maintains the good metrics of the original placement without generating new buffer violations (defined later).

We apply a simple and conservative approach to insert buffers into critical nets defined based on the buffer critical length. (The goal is to evaluate whether our approach facilitate the convergence of buffer insertion. If the goal is to minimize the number of buffers inserted, a more sophisticated buffer insertion algorithms, [9] for example, may be applied instead.) The buffer critical length L_{crit} is defined as the minimum net length above which inserting optimal-sized and optimal-located buffer can reduce the delay compared to the unbuffered net [12]. A net has buffer violation if its HPWL is larger than $1.4 \times L_{crit}$. We define a net to be buffer critical if its HPWL is larger than $1.2 \times L_{crit}$. By inserting buffers, we decompose a critical net into several smaller nets and remove the buffer violation of the net. A couple of minor changes are made to the placement flow. We iteratively apply buffer insertion and floorplan sizing (i.e., floorplan management without the detailed placement step) to remove all the buffer violations. Again, we maintain the relative amount of white space by expanding the core region after buffer insertion. We enforce an upper bound constraint ($1.4 \times L_{crit}$) on the length of every net during the detailed placement step.

We apply our buffer insertion approach at 45 nm technology on IBM benchmarks, assuming that interconnects do not scale accordingly. L_{crit} at 45nm technology is estimated to be 235 micron [12].

In our experiments, at most two iterations of buffer insertion and floorplan sizing are needed for the buffer insertion flow to converge. To evaluate our floorplan management approach, we compare our solutions with solutions obtained from running CPT and Dragon on the buffered netlists. Experimental results are shown in Table 4. We show the number of buffer violations ('bv') of initial placements, placements right after buffer insertion, and final placements obtained by the three approaches. These results show that our approach can converge to a placement without any buffer violation, whereas applying CPT or Dragon on buffered netlist may generate new buffer violations. Note that buffer violations are defined by HPWL and not by routed wirelength. However, we expect the results to be similar. Table 4 also shows the statistics of cell displacements and net length changes of our approach. The average cell displacements and net length changes of our approach are 1.0% and 0.4% of core region bounding box on the average over all benchmarks. Moreover, Table 4 also shows the routing results of the placement solutions obtained by the three approaches. Our approach produces placements with 9.3% and 4.7% shorter routed wirelengths than CPT and Dragon respectively, and with comparable routability. In fact, with the additional numbers of buffers added ('bufs', normalized to the total cell count of the initial placement) and the increases in the core areas ('A'), the wirelengths after our approach are even smaller in most cases compared to the original placements (see Table 1). In summary, our incremental placement approach can preserve the stability of placements with a shorter runtime by avoiding another run of a placer.

5. CONCLUSION

In this paper, we propose an incremental placement method by floorplan management. We apply this approach to accommodate changes due to gate sizing and buffer insertion during placement process. We can handle a wide range of incremental changes to the layout and netlist due to gate sizing (1%-100%) and as many as 40% additional inserted buffers. Compared to other techniques, our approach produces placements with shorter routed wirelengths and better or comparable routability. Most important, our incremental approach can maintain the stability of placements.

6. REFERENCES

[1] S. Adya, I. Markov, and P. Villarrubia. On whitespace and stability in mixed-size placement and physical synthesis. In *Proc. Int. Conf. on Computer Aided Design*, pages 311–318, 2003.

[2] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Optimal partitioners and end-case placers for standard-cell layout. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(11):1304–1313, 2000.

[3] T. Chan, J. Cong, T. Kong, J. Shinnerl, and K. Sze. An enhanced multilevel algorithm for circuit placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 299–306, 2003.

[4] Y.-H. Chan, P. Kudva, L. Lacey, G. Northrop, and T. Rosser. Physical synthesis methodology for high performance microprocessors. In *Proc. Design Automation Conf*, pages 696–701, 2003.

[5] C.-C. Chang, J. Cong, and M. Xie. Optimality and scalability study of existing placement algorithms. In *Proc. Asia South Pacific Design Automation Conf.*, pages 621–627, 2003.

[6] W. Chen, C. Hsieh, and M. Pedram. Simultaneously gate sizing and placement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(2):206–214, 2000.

[7] J. Cong and M. Sarrafzadeh. Incremental physical design. In *Proc. Int. Symp. on Physical Design*, pages 34–42, 2000.

[8] A. Khatkhate, C. Li, A. Agnihotri, M. Yildiz, S. Ono, C.-K. Koh, and P. Madden. Recursive bisection based mixed block placement. In *Proc. Int. Symp. on Physical Design*, pages 84–89, 2004.

[9] L. P. P. P. van Ginneken. Buffer placement in distributed rc-tree networks for minimal Elmore delay. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 865–868, 1990.

[10] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden. Routability-driven placement and white space allocation. In *Proc. Int. Conf. on Computer Aided Design*, 2004.

[11] P. Saxena and B. Halpin. Modeling repeaters explicitly within analytical placement. In *Proc. Design Automation Conf*, pages 699–704, 2004.

[12] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpartrick. Repeater scaling and its impact on CAD. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(4):451–463, 2004.

[13] G. Stenz, B. M. Riess, B. Rohfleisch, and F. M. Johannes. Timing driven placement in interaction with netlist transformations. In *Proc. Int. Symp. on Physical Design*, pages 36 – 41, 1997.

[14] M. Vujkovic, D. Wadkins, B. Swartz, and C. Sechen. Efficient timing closure without timing driven placement and routing. In *Proc. Design Automation Conf*, pages 268–273, 2004.

[15] Z. Xiu, J. D. Ma, S. M. Fowler, and R. A. Rutenbar. Large-scale placement by grid-warping. In *Proc. Design Automation Conf*, pages 351–356, 2004.

[16] X. Yang, B.-K. Choi, and M. Sarrafzadeh. Routability-driven white space allocation for fixed-die standard-cell placement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(4):410–419, 2003.