Low Power Clock Buffer Planning Methodology in F-D Placement for Large Scale Circuit Design*

Yanfeng Wang[a], Qiang Zhou[a], Yici Cai[a], Jiang Hu[b], Xianlong Hong[a], Jinian Bian[a]

[a]Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology
[a]Tsinghua University, Beijing, 100084, China
[b]Department of Electrical Engineering, Texas A&M University, College Station, TX 77843, U.S.A.
Email: [a]wangyanfeng05@mails.tsinghua.edu.cn, [b]jianghu@ee.tamu.edu

Abstract— Traditionally, clock network layout is performed after cell placement. Such methodology is facing a serious problem in nanometer IC designs where people tend to use huge clock buffers for robustness against variations. That is, clock buffers are often placed far from ideal locations to avoid overlap with logic cells. As a result, both power dissipation and timing are degraded. In order to solve this problem, we propose a low power clock buffer planning methodology which is integrated with cell placement. A Bin-Divided Grouping algorithm is developed to construct virtual buffer tree, which can explicitly model the clock buffers in placement. The virtual buffer tree is dynamically updated during the placement to reflect the changes of latch locations. To reduce power dissipation, latch clumping is incorporated with the clock buffer planning. The experimental results show that our method can reduce clock power significantly by 21% on average.

I. Introduction

In large scale ultra-deep submicron VLSI designs, clock network construction has become increasingly challenging due to many problems such as timing, power consumption, power supply noise and tolerance to process variations. Buffers are essential in a clock network as they not only improve signal slew rate, but also greatly affect clock skew, delay and clock network power.

Various clock buffering approaches have been developed in the past twenty years. The algorithm in [1] inserts the same number of buffers in each source-to-sink path and equalizes the capacitance driven by each buffer in an effort to reduce the skew sensitivity to process variations. The work of [2] attempts to reduce skew sensitivity by optimizing the number of buffer levels, buffer sizes and wire widths in a dynamic programming based approach. In [3], a three stage optimization algorithm is proposed to minimize the clock skew and delay and achieves great delay improvement. By making use of tolerable skew constraints, [4] employs a balanced buffer insertion scheme to obtain the minimal wire widths, which can result in minimal wiring capacitance and dynamic power dissipation. The work of [5] uses Lagrangian Relaxation to simultaneously minimize delay, power, and area with very low skew and sensitivity. Considering the large size of clock buffers, [6] generalizes the concept of merging segment in DME algorithm to merging block, which is a region reserved for clock buffers. Then, a graph theory based method is employed to place buffers in their merging blocks so that the overlaps between buffers can be removed. In [13] and [14], new algorithms are proposed to perform buffering and clock routing simultaneously.

However, all these previous works perform clock buffering after cell placement, and suffer from a common drawback. In modern IC designs, people tend to use huge clock buffers, which are relatively insensitive to variations. Therefore, it is often difficult to find large open space for such huge buffers. To avoid the overlap with logic cells, clock buffers are often placed far from ideal location. As a result, both power dissipation and timing are degraded. Traditionally, ECO methods are always used to deal with such problem, but it takes too many iterations to reach convergence. Especially in large scale designs, searching space for the buffers would slow down convergence considerably. To solve this problem efficiently, a clock buffer planning should be performed to reserve adequate space for the buffers at desired locations. If the reserved space is far from the ideal locations, the clock net connecting the latches and buffers will become much longer, and this leads to an increasing power dissipation on the clock net as well as other timing and variation problems. In Fig.1, a small example is depicted to illustrate the effect of buffer planning. In Fig.1(a), without buffer planning, the buffers are inserted to the location where overlaps are relative

*This work was supported partly by National Science Foundation of China (90407005, 90607001 and 60776026), and project of 3-D floorplanning and placement of Intel Corporation

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

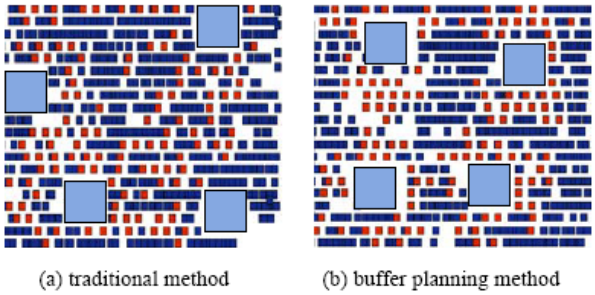


Fig. 1. Small blue rectangles represent signal cells, red rectangles for latches and large rectangles for clock buffers. Placement (a) is from tradition method which may place clock buffers far from ideal locations. Placement (b) from buffer planning method which can places the clock buffers in the ideal locations.

small, but this will increase the clock net a lot. Whereas in Fig.1(b), by using buffer planning and latch clustering scheme, enough large spaces for clock buffers are reserved at proper location in accordance with the latch distribution. Then the clock buffers will be inserted at such spaces which could reduce the clock wire length.

Power dissipation is now becoming the most challenging issue in modern large scale IC designs and has become a bottle-neck constraint for circuit performance growth. To alleviate power consumption, a latch clumping technique considering buffer and latch placement is proposed in [7]. It is based on the observation that most of power is dissipated in the lower level subtrees which are driven by low level clock buffers that directly drive the latches. The low level clock buffer is called local clock buffer (LCB) in [7][1]. The power consumed on the LCB level could be significantly decreased by placing the latches closer to their corresponding LCBs. Meanwhile, constraining all the latches around LCBs could have positive effect on timing due to clock skew reduction. Previous work [8] and [9] have dealt with the latch distribution problem for power reduction under skew constraint. In [10], a placement methodology is introduced to improve skew tolerance to variations. However, none of these latch placement works considered clock buffers. In reality, a clock buffering solution is highly related to latch placement.

In this paper, we propose a clock buffer planning methodology for low power and large scale clock network design. To the best of our knowledge, there is no previous work on clock buffer planning yet. The proposed planning is integrated with cell placement and combined with latch clumping in order to achieve a low power design. Simultaneous buffer planning and latch clumping is difficult as both techniques may lead to negative side-effect to logic signal wire length. Unlike [7], which rigidly places all latches around LCBs, our method allows certain latch placement flexibility such that the side-effect to signal wire length is reduced. Therefore, our method can reach a nice tradeoff between latch clumping and signal wire length overhead. By employing an Dynamic Clock Tree technique, we can seamlessly integrate buffer planning, power optimization and placement. Hence, a low power clock network can be obtained with acceptable signal wire length overhead. Moreover, the computation convergence rate is better than traditional ECO method. Experimental results show that our method can reserve sufficient spaces for clock buffers at desired locations with a better convergence than traditional method. Due to our approach, the total power of the chip can be reduced by 21.6% on average. In addition, benefit on the timing of the subtrees driven by LCBs is also observed.

[1]Throughout this paper, we continue to use the term LCB for this kind of buffer.

II. Preliminary

A. Force-Directed Placement

Force-directed method [11] as a global placement technique uses Spread Force to reduce the cell overlaps and iterative quadratic optimizations to achieve the spread metric. FDP [12] is a stable force-directed placer which could produce high quality placements. Compared to the previous force-directed algorithms, it has introduced a new Boxplace Force to aid directly in the minimization of wire length. In this paper, our optimization for latches and buffers will be integrated into the FDP framework, thus by imposing the clock force and weighing the new clock force and the original force properly, we could achieve our optimization objective with good placement results and obtain an optimal tradeoff between clocking and the traditional objectives.

B. Buffered Clock Tree Design

Clock network delivers clock signal from the clock source to the clock sinks to control all these synchronous elements to work simultaneously. The most familiar clock network structure is a binary routing tree where the root node represents the clock source and the leaf nodes stand for the clock sinks. Typically, the clock tree is conceived in a bottom-up fashion by a merging and embedding process.

Clock buffer deployment has constituted an essential part in clock design. Usually, clock buffers are inserted at the merging nodes of the clock tree to generate local clock signals for the output subtrees. To keep the clock tree balanced, the clock buffers are often inserted level by level. Nowadays, people tend to use huge clock buffers, which are relatively insensitive to variations, but this makes the buffering work very difficult in large scale designs since original placement have to be significantly perturbed to make sufficient space for such large amount of clock buffers.

III. Proposed Approach

We propose a **Low Power Buffer Planning (LPBP)** method which is embedded in a cell placement algorithm. In addition to traditional placement objectives such as minimizing total wire length, the LPBP method should lead to a cell placement such that certain spaces are reserved for clock buffers and latches are clumped in clusters.

In order to achieve the objective of LPBP, we need to anticipate how much buffer area will be required and where they will be inserted in clock network layout. Such anticipation can guide the cell placer to reserve the proper amount of space at desired locations. To model the clock buffers in placement, we propose a fast grouping method called Bin-Divided Grouping which groups a set of clock sinks (the clock sinks can be either latches or clock buffers) into several clusters. For each cluster, a virtual clock buffer is modeled to drive all the clock sinks in this cluster. Starting from the latches, the grouping is performed recursively to construct a virtual buffer tree. This buffer tree is an abstract tree as a part of the netlist for the cell placer to anticipate where and how much clock buffer area is needed. To reflect the changes of latch locations during the placement, we use a Dynamic Clock Tree Rebuilding (DCTR) technique such that the buffer tree is updated in the placement iterations. Such dynamic update ensures that the buffer model is in accordance with placement changes. To further reduce power dissipation, we impose a Contract Force on the latches to achieve latch clumping. The whole process is carried out in a FDP framework.

A. Virtual Buffer Tree (VBT)

Almost all of the existing works on clock buffering [1, 3] are carried during clock network layout. Thus, they are based on well-defined latch locations and clock tree routings. Since our clock buffer planning is embedded within cell placement, the latch locations have not been finally decided yet and there is no clock routing at all. Hence, the clock buffer planning is much difficult than the traditional clock buffering. The buffer planning has to be based on some definite information, which is limited. First, we assume that if the total capacitance of a group of clock sinks is greater than certain threshold $C_{threshold}$, we need to insert a clock buffer to drive them. Ideally, it would be better to take wire capacitance into account. However, the locations of clock sinks keep changing during the placement. Such changes make clock routing wire length intractable. Second, we use a balanced clock tree design in the planning and attempt to estimate the buffers of multiple levels in the tree. Third, the clock sinks here can be either latches or clock buffers based on the level-by-level buffering.

For a set of clock sinks, if their total capacitance is equal to C_{total}, the number of buffers required to drive these sinks can be roughly estimated as $K = \lceil C_{total}/C_{threshold} \rceil$. We cluster the clock sinks into K groups with about the same capacitance. Then, one clock buffer is allocated to drive each group of clock sinks. The clock sinks in the same group should be physically close to each other. We formulate the **Clock Sink Grouping problem** as follows.

Given a set of clock sinks $\mathcal{V} = \{v_1, v_2, ...v_n\}$ and load capacitance c_i for each sink $v_i \in V$, group them into K subsets $\{V_1, V_2, ...V_K\}$. Let n_k denote the number of clock sinks in V_k, C_k denote the total capacitance in V_k, set $\mathcal{C} = \{C_1, C_2, ...C_K\}$, g_k denote the COG (center of gravity) point of V_k. Define the cost of w_k as

$$w_k = \sum_{v_j \in V_k} dist(g_k, v_j) \tag{1}$$

in which $dist(g_k, v_j)$ stands for the Euclidean distance between point g_k and sink v_j. The objective function, which is the total cost, is defined as:

$$W = \alpha \sum_{k=1}^{K} w_k + \beta\sigma(\mathcal{C}) \tag{2}$$

where α and β are constant weights and $\sigma(\mathcal{C})$ represents the standard deviation of set $\mathcal{C}$. The objective of the group is to minimize the overall cost W.

Obviously, this problem is more difficult than graph partitioning, which is an NP-complete problem. Therefore, we will find a heuristic approach to solve this problem. In realistic clock network designs, people often use balanced tree structure for the sake of skew management and variation tolerance. Based on such balanced structure, a simple and fast grouping method named Bin-Divided Grouping (BDG) algorithm is presented in table I. BDG cuts the entire layout area into several bins with about the same total capacitance. By pre-sorting the clock sinks physically, BDG can achieve the computational complexity of $O(n \log n)$ where n is the number of clock sinks. Most of the computation time is consumed on the sorting and only linear time is spent at the dividing process. We can see that the bins generated by our algorithm do not intersect each other and they are arrayed in a brick structure as is shown in Fig.2.

By using BDG, we can model the buffers level by level until we reach the level where the total capacitance of the clock sinks is smaller than $C_{threshold}$. These virtual buffers are related with each other through a tree tree structure, called "virtual buffer tree". In a virtual buffer tree, the leaf nodes represent the latches, the internal nodes represent clock buffers and the edge connecting two nodes represents a parent-child relationship. Unlike the physical routing tree in the clock routing stage, the virtual buffer tree is an abstract tree and only exists in placement stage as a part of netlist. It is also constructed by recursive merging starting from the leaf (latch) level. First, the latches are grouped into subsets using BDG. For

TABLE I
BIN-DIVIDED GROUPING ALGORITHM

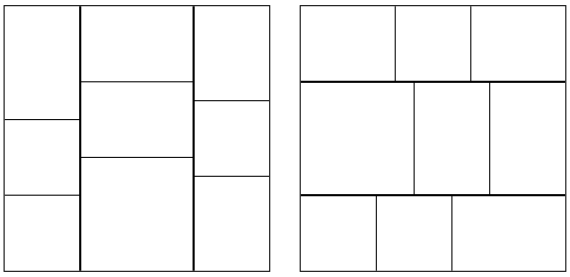
Input:	a set of clock sinks $\{v_1, v_2, ...v_n\}$ with capacitance $\{c_1, c_2, ...c_n\}$ with coordinates $\{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$ with the pre-processed number K with $K = p \times q$
output:	K clock sink subsets $\{C_1, C_2, ...C_K\}$
1.	**begin**
2.	set $threshold = (\sum_{i=1}^{n} c_i)/p, B = \{v_1, v_2, ...v_n\}$
3.	for i = *1* to p
4.	initial B_i as a empty set
5.	while total capacitance in B_i ¡ *threshold*
6.	find v_k with the smallest x coodinate
7.	$B_i = B_i \bigcup v_k$ and $B = B - v_k$
8.	divide B_i to q sub-bins according to y-coordinate with the same manner
9.	compute the total cost of these sub-bins as *costx*
10.	redo the whole divide process but first with y-coordinate second with x-coordinate and compute total cost as *costy*
11.	choose the cost-smaller results as the final sub-bins
12.	group the clock sinks in $B_1...B_K$ as $C_1...C_K$
13.	**end**

each subset, a merging node is inserted at its COG (center of gravity) and serves as the parent node of the sinks in the subset. Next, the newly generated merging nodes are treated as virtual clock sinks and the merging procedure is applied again on them. Such merging is continued recursively till we reach the level which buffer insertion is not needed. Finally, we generate a single node as the parent of the clock sinks on the top level. Apparently, most of the construction time is consumed at the grouping work, thus the computational complexity of constructing the virtual buffer tree is also $O(n \log n)$.

The main purpose of using virtual buffer tree is to let cell placement have certain prediction on the need of clock network. Constructing virtual buffer trees in placement stage has three advantages. First, we can choose any level of the tree and clump the subtrees rooted in that level to reduce the local clock wire length. Second, by pre-assigning the locations of merging nodes, the descendant latches of the merging nodes can be pulled towards some planned locations. Third, by continually adjusting the positions of the merging nodes according to the placement changes of latches, buffer space can be reserved at desired locations.

B. Interaction Between VBT and Placement

To integrate the VBT (Virtual Buffer Tree) into FDP (Force-Directed Placement), we need to construct the VBT at certain step of placement. Then, the placement continues with an altered netlist. The effect of the VBT depends on the stage where it is built. If the VBT is constructed at an early placement stage, the influence to the convergence of placement algorithm is small but its impact on placement solution is large. Besides, the latches clustered in the same group by the VBT may be shifted far from each other due to the instability of latch placement in early stages. Consequently, the clock wire length may be unnecessarily large. Of course, we can force the latches in the same group to remain close to each other throughout the placement. But, the wire length of logic signals may increase badly. On the other hand, if the VBT is constructed in late stages, the placement is relatively stable and therefore the effect of the VBT is close to the method of ECO methodology. As such, the impact to signal wire length is small but the convergence of the placement is largely slow down. Therefore, the stage for the VBT construction needs to be judiciously selected such that a desired tradeoff between the two extremes can be reached.



(a) x-first cut (b) y-first cut

Fig. 2. The brick structure of the bins, (a) is generated by x-first cut, (b) is generated by y-first cut

The topology of VBT is a prediction to the subsequent clock network design. However, when the locations of latches changes in the placement, such prediction should be tuned accordingly. Otherwise, the placement may proceed with a poor prediction and thereby lead to a solution hindering rather than helping the clock network design. If the VBT built at an earlier stage deviates does not fit the latch locations in a later stage, the VBT should be rebuilt for a better fidelity. We call this technique as Dynamic Clock Tree Rebuilding (DCTR). DCTR has two major advantages. First, the tree topology with good conformity to the placement trend will be inherited in the later version of rebuilt topology. Second, early generated topology, which becomes discrepant later, still can be refined. Such dynamic refinement can make sure that the VBT is always consistent with the latest latch placement. By using DCTR, the adaptive capability VBT is remarkably improved. Thus, the aforementioned tradeoff becomes easier to be achieved.

Although DCTR can help to improve both solution quality and convergence rate, too frequent update on VBT does not help much but causes unnecessary computation increase. Usually, the VBT is rebuilt after several iterations when certain spreading criterion is reached. The computation complexity after using DCTR becomes

$O(c \cdot n \log n)$ where c represents the times of rebuilding.

In order to make impact of VBT more smooth, we suggest gradually changing sizes for the virtual buffers. If we use the actual size for the buffers when the VBT is first built, the huge size often gives rise to a sudden intrusion to the cell placement. Moreover, when the VBT is rebuilt in early placement stages, huge clock buffers may cause large perturbation to the placement and slow down the convergence. Therefore, we artificially use a very small size for the virtual buffers at the beginning and then gradually increase the size throughout the iterations. The gradual increase of buffer size makes sure that newly generated virtual buffers from rebuilding VBT have sufficient space, which are released from the old VBT. Eventually, the size increases to the actual physical size of clock buffers in late stages. The virtual buffers with actual size can effectively reserve space for subsequent clock network design. By using this scheme, the interaction between the VBT/DCTR and cell placement becomes much more smooth.

C. Latch Clumping

In addition to the clock buffer planning, clumping latches close to each other also facilitates low power network. Since latch clumping [8] is carried out in cell placement, we integrate it with the clock buffer planning for further power reduction. The latches are imposed with contract forces which point to their parent nodes on the buffer tree, i.e., LCBs. We combine such contract force with the original forces (Spread Force and Boxplace Force) in FDP. This is illustrated in Fig.3(b). However, the latch clumping may reduce the space reserved for clock buffers. Therefore, we impose a Center Force on the buffers which pulls them towards the COG of the group they drive. This center force is also combined with the original forces in FDP. Since the virtual buffers are not connected to any signal cells, the placement of virtual buffers does not have direct effect on the signal wire length. Hence, we delete the Boxplace Force on the buffers and only use Spread Force together with Center Force as shown in Fig.3(a).

A key difference between our approach and previous work on latch clumping [8, 9] is the handling on clock buffers. The previous works [8, 9] do not consider the impact on clock buffers in the latch clumping. In contrast, our approach seamlessly integrates the latch clumping with the clock buffer planning. More specifically, our latch clumping is always consistent with the clock sink clustering in DCTR. Therefore, we do not need to pull latches through long distances for the clumping. The gradual increase of the virtual buffer size also makes the clumping and clustering smooth. All of these improve the convergence and minimize negative impact to signal wire length.

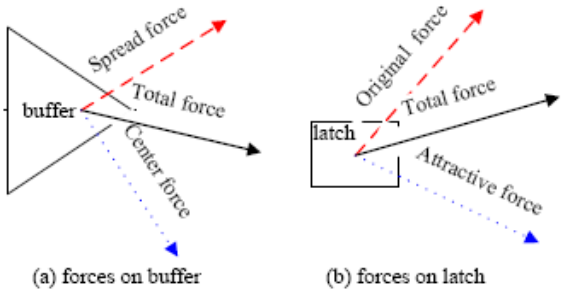


Fig. 3. clock forces imposed on buffers and latches, (a) represents forces on buffers and (b) represents forces on latches

TABLE II
CONFORMITY OF RESERVED SPACE TO BST ROUTER

Design	Cell Num	Latch Num	Scheme	LCB Num	LCB Diff	High Num	High [0,1]	High [1,2]	High [2,max]
ben01	12752	1872	ECO	64	9 (14%)	20	18	1	1 (5%)
			LPBP		9 (14%)		20	0	0 (0%)
ben02	19601	2898	ECO	90	10 (11%)	28	25	3	0 (0%)
			LPBP		9 (10%)		23	5	0 (0%)
ben03	23136	3420	ECO	110	10 (9%)	38	35	3	0 (0%)
			LPBP		11 (10%)		35	3	0 (0%)
ben04	27507	4077	ECO	132	11 (8%)	38	30	6	2 (5%)
			LPBP		13 (10%)		34	3	1 (3%)
ben05	29347	4221	ECO	144	13 (9%)	47	35	11	1 (2%)
			LPBP		16 (11%)		41	5	1 (2%)
ben06	32498	4842	ECO	156	12 (8%)	56	44	10	2 (4%)
			LPBP		12 (8%)		45	8	2 (4%)
ben07	45926	6840	ECO	225	15 (7%)	70	49	19	2 (3%)
			LPBP		18 (8%)		59	8	3 (4%)
ben08	51309	7650	ECO	256	17 (7%)	84	66	16	2 (2%)
			LPBP		16 (6%)		75	9	0 (0%)

IV. Experimental Results

The test cases employed in our experiments is based on ISPD02 placement benchmark. Since our clock buffer planning method is design for large circuits, ISCAS89 sequential circuits are not suitable due to their small size. We choose 8 circuits from ISPD02 benchmark suite and random specify some of their cells as latches. These circuits are named as ben01-ben08. The number of cells and the number of latches are listed in the second and the third column of Table II, respectively. We implemented and compared the following two methods:

1. ECO: a virtual buffer tree is constructed only in the late stage of placement process. This is very similar to traditional ECO methodology where additional placement changes are made to find space for clock buffers.

2. LPBP (Low Power Buffer Planning): this is our proposed clock buffer planning methodology with integrated latch clumping.

First we measure whether the reserved buffer space fits for the actual clock router. We use BST [15] as the clock

TABLE III
COMPARISON BETWEEN ECO AND LPBP ON SIGNAL WIRE LENGTH, CLOCK WIRE LENGTH, CPU TIME, POWER AND TIMING

Design	Scheme	Signal WL	Cpu Time	Clock WL(e03)	Power (w)	LCB Delay(ps)	LCB Skew(ps)
ben1	ECO	1.93	22	4.55	1.35	462.24	149.63
	LPBP	2.03	14	2.92	1.06	352.49	78.96
	Impr.	-5.3%	36.4%	35.9%	21.8%	23.7%	47.2%
ben2	ECO	4.21	54	6.75	1.96	425.47	88.72
	LPBP	4.36	52	4.10	1.54	359.69	78.98
	Impr.	-3.5%	3.8%	39.2%	21.4%	15.5%	11.0%
ben3	ECO	5.98	42	8.16	2.42	476.38	152.25
	LPBP	6.15	38	5.19	1.90	372.07	96.14
	Impr.	-2.8%	9.5%	36.3%	21.3%	21.9%	36.9%
ben4	ECO	6.87	67	9.94	2.93	480.10	164.14
	LPBP	7.08	52	6.46	2.31	375.23	96.52
	Impr.	-3.1%	22.4%	35.0%	21.1%	21.8%	41.2%
ben5	ECO	11.29	73	10.24	3.07	405.45	103.44
	LPBP	11.57	56	6.35	2.39	348.07	74.19
	Impr.	-2.5%	23.3%	38.0%	22.1%	14.2%	28.3%
ben6	ECO	6.29	90	10.07	3.31	430.36	133.94
	LPBP	6.46	60	6.80	2.63	349.76	76.61
	Impr.	-2.7%	33.3%	32.5%	20.5%	18.7%	42.8%
ben7	ECO	10.72	91	15.68	4.78	418.90	116.35
	LPBP	10.98	65	9.48	3.72	345.30	70.73
	Impr.	-2.6%	28.6%	39.5%	22.1%	17.6%	39.2%
ben8	ECO	11.81	74	17.23	5.38	426.96	124.75
	LPBP	11.59	57	10.23	4.16	346.66	86.43
	Impr.	1.9%	23.0%	40.7%	22.7%	18.8%	30.7%
	Avg.Impro.	-3.1%	20.0%	37.1%	21.6%	19.0%	34.7%

router to construct a zero skew clock routing tree. If the LCB estimated from BST is within or very close to the space reserved by virtual buffers in the planning, we deem that the buffer space conforms to the BST totally. For high level clock buffers, we separately count the number of reserved spaces which are positioned within one-buffer size, within two-buffer size but beyond one-buffer size, and beyond two-buffer size to the ideal buffer locations estimated from BST. From the results presented in Table II, one can see that the ratio of LCBs which do not cover the ideal buffer locations is very small for each benchmark. Meanwhile, the high level buffer spaces which are positioned beyond two-buffer size distance to the ideal buffer locations are very few. Also, by comparing the results obtained by ECO and LPBP, we can see the latch clumping of LPBP does not degrade the quality of reserved space locations.

Table III shows the comparison between ECO and LPBP on wire length, power and timing. Compared to ECO, LPBP can reduce CPU time by 20% on average through improving convergence rate. The increase on signal wire length from LPBP is very limited and is only 3.1% on average. The clock wire length is obtained from BST results, which indicate averagely 37.1% reduction from our LPBP. We use the model proposed in [16] to estimate the power consumed on clock net. One can see that LPBP can achieve 21.6% power reduction on average. Also, we measured the clock delay and clock skew at the LCB level by the Elmore Delay model. Averagely 19% reduction of LCB level delay and 34.7% reduction of skew are observed from our LPBP method.

V. Summary and Conclusions

In this paper, we propose a clock buffer planning methodology which can let cell placement reserve sufficent space for clock buffers at desired locations. The planning can facilitate low power clock network design. The planning is further integrated with latch clumping for further power reduction. Experimental results show significant power decrease due to our approach.

References

[1] S. Pullela, N. Menezes, J. Omar and L. T. Pillage, "Skew and Delay Optimization For Reliable Buffered Clock Trees", *in Proc. ICCAD*, pp. 556-562, 1993.

[2] Chung J., Cheng C. K. "Optimal buffered clock tree synthesis", *in Proc. ASIC*, pp. 130-133, 1994.

[3] X. Zeng, D. Zhou, and W. Li, "Buffer insertion for clock delay and skew minimization", *in Proc. ISPD*, pp. 36-41, 1999.

[4] Joe G. Xi, Wayne W. M. Dai "Buffer insertion and sizing under process variations for low power clock distribution", *in Proc. DAC*, pp. 491-496, 1995.

[5] Chung-Ping Chen, Yao-Wen Chang, and D. F. Wong, "Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation", *in Proc. DAC*, pp. 405-408, 1996.

[6] Y. P. Chen and D. F. Wong, "An Algorithm for Zero-Skew Clock Tree Routing with Buffer Insertion", *in Proc. EDTC*, pp. 66-71, 1996.

[7] Ruchir Puri, Ruchir Puri and Subhrajit Bhattacharya, "Keeping Hot Chips Cool", *in Proc. DAC*, pp. 285-288, 2005.

[8] Yongseok Cheon, Pei-Hsin Ho, Andrew B. Kahng, Sherief Reda, Qinke Wang, "Power-Aware Placement", *in Proc. DAC*, pp. 795-800, 2005.

[9] Yongqiang Lu, C. N. Sze, Xianlong Hong, Qiang Zhou, Yici Cai, Liang Huang, Jiang Hu, "Navigating Registers in Placement for Clock Network Minimization", *in Proc. DAC*, pp. 176-181, 2005.

[10] G. Venkataraman, J. Hu, "A placement methodology for robust clocking," *in Proc. International Conference on VLSI Design*, pp. 881-886, 2007.

[11] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning", *in Proc. of DAC*, pp. 269C274, 1998.

[12] K. P. Vorwerk, A. Kennings, and A. Vannelli, "Engineering details of a stable force-directed placer", *in Proc. ICCAD*, pp. 573-580, 2004.

[13] Rishi Chaturvedi and Jiang Hu, "Buffered clock tree for high quality IC design", *in Proc. ISQED*, pp. 381-386, 2004.

[14] I-Min Liu, Tan-Li Chou, Adnan Aziz, D. F. Wang, "Zero-Skew Clock Tree Construction by Simultaneous Routing, Wire Sizing and Buffer Insertion", *in Proc. ISPD*, pp. 33-38, 2000.

[15] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and Steiner routing", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 3(3):341-388, July 1998.

[16] W. Liao and L. He, "Full-chip Interconnect Power Estimation and Simulation Considering Concurrent Repeater and Flip-flop Insertion," *in Proc. ICCAD*, pp. 574-580, 2003.