A Buffer Planning Algorithm with Congestion Optimization[τ]

Song Chen[1], Xianlong Hong[1], Sheqin Dong[1], Yuchun Ma[1], Yici Cai[1],
Chung-Kuan Cheng[2], Jun Gu[3]
[1]Dept. of Computer Science & Technology, Tsinghua Univ., Beijing, China
[2]Dept. of Computer Science and Engineering, Univ. of California, San Diego USA
[3]Dept. of Computer Science, Science & Technology University of HongKong

Abstract - This paper studies the problem of buffer planning for interconnect-centric floorplanning. We devise a congestion-driven buffer insertion algorithm, in which single-pair shortest-path model is used to compute optimal buffer locations and simultaneously to preserve the monotonicity of routing paths. Congestion estimation is achieved by an approach of probabilistic analysis. In order to get more buffers inserted, on the basis of a rough estimation of buffer locations, some channels determined by the boundaries of circuit block are inserted, while the topology of the placement keeps unchanged. Furthermore, we change the distribution of the dead space among blocks to optimize the time closure and routing congestion. The performance of the chip can be improved greatly on penalty of a small area usage. The effectiveness of the proposed algorithm has been demonstrated by the experimental results.

I. INTRODUCTION

As the VLSI circuits are scaled into nanometer dimensions and operate in gigahertz frequencies, interconnect design and optimization has become critical. To ensure the timing closure of design, interconnects must be considered as early as possible in the design flow.

Buffer insertion is an effective technique to reduce the interconnect delay. While the Elmore delay of a long wire grows quadratically in terms of the length of the wire, buffer insertion properly results in a linear delay increase due to the length of the wire. The number of buffers needed to achieve timing closure continues to increase with decreasing feature size. Buffers must be planned early in the design because they will take up silicon resources. It is very useful that a good planning of the buffer positions can be obtained in the placement stage to favor later routing stages.

A. Previous Work

J. Cong et al[2] introduces the concept of *feasible region*, which is used to generate buffer bocks. Sarkar et al[5] adds the notion of independence to feasible region and tries to improve the routing congestion. These two papers give the basic idea of *Feasible Region*, on which they proposed the buffer planning algorithm. But both of their methods take complex scanning to obtain the feasible buffer insertion sites, and [5] decomposes multi-pin nets by the star model, which will cause a very over-estimated congestion. Tang and Wong[8] proposes an optimal algorithm based net flow to assign buffers to buffer blocks assuming only one buffer for each net. F.F. Dragan[9][10] allocates buffers to existent buffer blocks by the multi-commodity flow-based approach. Alpert [11] makes use of tile graph and dynamic programming to perform buffer block planning. They assume that buffers be allowed to be inserted inside macro blocks. S. Chen [15] optimizes the distribution of the dead space to improve the number of nets that satisfy delay constrains and buffer insertion is modeled as max cardinality matching problem, but the monotonicity of routing paths cannot be retained. Y. Ma [18] and K.W.C. Wong [19] integrate floorplanning with buffer planning and congestion analysis. Although buffers are mostly inserted into dead space and some of previous works consider routability, none of them optimize buffer insertion and the routing congestion by changing the distribution of the dead space in the results of the floorplanning.

B. Our Contributions

This paper proposes approach to optimize the congestion and time closure by changing the distribution of the dead space in the placement, and devised a congestion-driven buffer insertion algorithm, in which a single-pair shortest--path model is used to compute optimal positions for buffers, and simultaneously to preserve the monotonicity of the routing paths. As a pre-processing, the chip is dissected into rooms in each of which at most one block is located, and in order to get more buffers inserted, on the basis of a rough estimation of buffer location, some channels determined by the boundary of circuit block are inserted into rooms that are entirely held by blocks, while the topology of the placement keeps unchanged. The performance of the chip can be improved greatly on penalty of a small area usage. The effectiveness of the proposed algorithm has been demonstrated by the experimental results.

Instead of star model that used in [2], [5], [15], and [18], *Minimum Spanning Tree (MST)* model is used to decompose multi-pin nets. In order to achieve the congestion estimation, an accurate congestion model [16] is adopted.

Given the results of the floorplanning, as a post-layout processing, our algorithm can achieve buffer planning and improve further the routing congestion of the chip.

The rest of the paper is organized as follows. Section II gives the problem definition, introduces the calculation of independent feasible region and gives a brief review of the redistribution of the dead space. The method of channel allocation is shown in section III. The congestion driven buffer insertion algorithm is discussed in section IV. Section V involves the congestion optimization based on the redistribution of dead space. Section VI and section VII give the experimental results and conclusion, respectively.

[τ] This work is supported by the National Natural Science Foundation of China 60121120706 and National Natural Science Foundation of USA CCR-0096383, the National Foundation Research(973) Program of China G1998030403, the National Natural Science Foundation of China 90307005, NSFC&HK GRC joint Grant No. 60218004 and 863 Hi-Tech Research & Development Program of China 2002AA1Z1460

II. PRELIMILARY

A. Problem Definition

In this paper, we concentrate on the buffer planning problem: Given an initial placement/floorplan and timing constraints for each net, we try to determine the number and locations of buffers for each net to achieve the timing closure of design, and take the routing congestion considered. The buffers are inserted into the dead-spaces and channels that are allocated based on a roughly estimation of the distribution of the buffers.

B. Independent Feasible Region

The concept of *independent feasible region (IFR)* is introduced for buffer insertion [5]. The *IFR* for a buffer *b* is the maximum region where *b* can be located such that by inserting buffer *b* into any location in that region, the net delay constraint can be satisfied, assuming that the other buffers of that net are also located within their respective independent feasible regions.

The minimum number of buffers to meet the delay constraint T_{req} for an interconnect of length *l* is

$$k_{\min} = \left\lceil \frac{-K_5 - \sqrt{K_5^2 - 4K_4K_6}}{2K_4} \right\rceil \qquad (1)$$

where

$$K_4 = R_bC_b + T_b$$
$$K_5 = (rC_b + cR_b)l + T_b + R_dC_b + R_bC_l - T_{req} - \frac{r}{2c}(C_b - C_l)2 - \frac{c}{2r}(R_b - R_d)^2$$
$$K_6 = \frac{1}{2}rcl^2 + (rC_l + cR_d)l + R_dC_l - T_{req}$$

The notation for the physical parameters of the interconnect and buffer we use in this paper is as follows:

- r wire resistance per unit length
- c wire capacitance pre unit length
- T_b intrinsic buffer delay
- C_b buffer input capacitance
- R_b buffer output resistance
- C_l sink capacitance;
- R_d driver resistance;
- l_n the length of source-sink net N (two-pin net)

The *IFR* of each buffer in a net can be calculated by means of the method developed in [5].

C. Redistribution of dead spaces

The dead-spaces are defined as the spaces within a placement that are not held by any circuit block. The chip can always be dissected into small rectangles, denoted as room, and there is at most one block in each room. All the rooms are not held entirely by the circuit blocks and there may be some empty room assigned no circuit block. In the former, the dead space is called *Detached Dead-Space (DDS)* [15], and the later type of dead-space is called *Attached Dead-Space (ADS)* [15].

The *DDS*s are moveless, while the distribution of *ADS*s can be changed. Through the topological representation of the placement/floorplan, we can respectively associate *ADS*s with some circuit blocks. Consequently, the distribution of dead-spaces can be changed by moving the circuit block within the room which is not entirely held.

III. CHANNELS ALLOCATION

Given a placement, the area of dead space is limited. Hence there are many buffers that may be inserted unsuccessfully to reduce the delay. It is not economical to reserve channels between each two adjacent blocks. In the work of [2] and [5], the channels are inserted during the buffer planning according to the necessaries, but the move of the blocks must change the pin position of nets, which maybe lead to a rough calculation of the number and locations of buffers. Therefore, we will first allocate channels before buffer planning. On the basis of estimating roughly the distribution of buffers, some vertical or horizontal channels are inserted into rooms held entirely by blocks, while the topology of the circuit keeps unchanged.

In general, the packing results are left-bottom compacted, we hence insert vertical channels into the left boundary or horizontal channels into bottom boundary of circuit blocks. For each block, if the center of the left boundary is located in the *IFR* of a buffer, we think that the buffer may be inserted into the left boundary nearby. Similarly, if the center of the bottom boundary is located in the IFR of a buffer, the buffer may be inserted into the bottom boundary nearby. For each block, we count the number of buffers possibly inserted the left boundary and bottom boundary nearby. Depending on the distribution of the buffers and the association among dead space and blocks, we determine whether or not the channels are inserted. The length of the channel is determined by the boundary of the block, and the width of channels depends on the number of possible inserted buffers.

IV. CONGESTION-DRIVEN BUFFER INSERTION

To perform congestion estimation, the chip is divided into 2-dimensional array of fixed-size grids. Hence, we can get the congestion information at every location of all over the floorplan. Because the dissection of the dead space for buffer insertion is separated from the dissection of the chip for routing congestion estimation, buffer resources has not to be considered during congestion estimating. Therefore, we can apply the traditional congestion model in our buffer planning algorithm. Fig.1 illustrated the congestion grids and the tiles buffer can be inserted. For simplicity, we think that a buffer lies in a grid if the center of the corresponding tile is located in the grid.

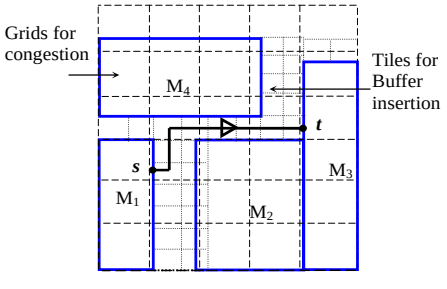

Fig.1 Grids for congestion estimation and tiles for buffer insertion

A. Congestion Estimation

The congestion model employed is essentially a two dimensional rectangular grid based probabilistic map assuming every multi-bend route of shortest Manhattan distance is feasible. The congestion of a routing grid (i, j) is defined as follows.

$cg^h(i, j)$ = Expected number of routes passing through grid (i, j) horizontally.

$cg^v(i, j)$ = Expected number of routes passing through grid (i, j) vertically.

To calculate the congestion numbers for the routing grid, we first compute the number of possible routes passing through every routing grid for each net with fixed source and fixed sink. Assuming that all these routes have equal probability, we obtain the probabilistic usage matrix which is similar to that in [16]. We compute the contribution of each net to the congestion matrices, $cg^h_s(i, j)$ and $cg^v_s(i, j)$. Thus, given a set of two pin nets, we can compute the congestion matrices and $cg^h(i, j)$ and $cg^v(i, j)$ as follows:

$$cg^h(i,j) = \sum_{s \in \{set\ of\ nets\}} cg^h_S(i,j)$$
$$cg^v(i,j) = \sum_{s \in \{set\ of\ nets\}} cg^v_S s(i,j) \quad (3)$$

On the other hand, with consideration of buffer positions, we define a sub-net as the segment of a net between two consecutive buffers, between the source and the first buffer, or between the last buffer and the sink. We compute the contribution of each subnet to the congestion matrices. The "set of nets" in formula (3) maybe include sub-nets in this situation.

The congestion is initialized by all the nets without consideration of buffers in the beginning of the buffer planning. And then the congestion estimation process is performed net by net. For each net, by solving a shortest path problem (discussed in sub-section *C*), we will select the best possible buffer locations for each net so that it is minimized that the sum of the congestion of the most congestive grid in the possible routing path of the sub-nets of the net. Following that, we will erase the congestion contribution of this net without consideration of buffers, and add the congestion contribution due to this net by breaking the net into a set of sub-nets consisting of all the source-buffer pair, buffer-buffer pairs and the buffer-sink pair along the route. The congestion information at each grid will be updated, which will affect the buffer insertion process of the other nets. The whole process is repeated until all the nets are routed and analyzed.

B. Multi-pins Nets Handling

In order to handle multi-pin nets, we need to decompose a multi-pin net into a set of two-pin nets. There are several methods to decompose a multi-pin net into two-pin nets such as star method, the MST method, or the rectilinear steiner tree (RST) method. The star-decomposed method is over-estimate congestion greatly because of overlapping net segments. MST may over-estimate the congestion. However, this conservative estimation will not affect the resultant packing significantly because the total length of an MST can be reduced at most by 6% to 9% by removing all the overlapping net segments to obtain a corresponding RST [12]. MST is a better choice for estimation purposes. As a result, we apply MST to handle multi-pin nets before our buffer planning algorithm. In our optimization algorithm, since the blocks are only moved in a local small area, we will not change the decomposition of the multi-pin nets.

C. Buffer Net with Congestion and Monotonicity considered

Definition 1 **Monotonicity** For a net N**,** if the routing path constrained by the inserted buffers (more than one) is a monotonic Manhattan path from the source to the sink, we say that the buffers satisfy **monotonicity** each other.

For each net with buffers inserted, to keep the routing path monotonic and select optimal buffer positions, we model the assignment problem of buffers to tiles as a single source to single sink shortest path problem, and the location of buffers are determined based on the shortest path.

For a net with n consecutive buffers b_1, b_2, …, b_n from source to sink, the s-t graph used for single-pair shortest-path model is constructed as follow.

1. *G = (V, E).*
2. *$V = V_1 \cup V_2 \cup \ldots \cup V_n \cup \{s, t\}$, where $V_i (i=1, ..., n)$ is the set of the candidate tiles of buffer b_i, s and t represent the source and the sink of the net respectively.*
3. *$E = E_1 \cup E_2 \cup \ldots \cup E_n \cup E_{n+1}$, where $E_1 = \{(s, v), v \in V_1\}$, $E_i = \{(u, v), u \in V_{i-1}, v \in V_i$, and u and v satisfy monotonicity$\}$, $i = 2, \ldots, n$, and $E_n = \{(v, t), v \in V_n\}$*

The cost of each edge in the graph G is assigned to the value of the most congestive grid located in the possible routing path from the source endpoint of the edge to the sink endpoint of the edge. In the graph G, each path from s to t represents one strategy of buffers insertion keeping the routing path monotonic. Therefore, we assigned the buffers of the net to tiles by finding the shortest path from *s* to *t*, which ensure the net with buffer inserted pass through grids with low congestion.

Fig.2 shows an example. For convenience, the tiles are not drawn. Two buffers b_1 and b_2 need to be inserted into the net to satisfy the delay constraint. As shown in the left of Fig.2, the light blue tiles are the candidate tiles of first buffer, and gray tiles are for the second buffer. And in the right side of the figure is the corresponding *s-t* graph which is constructed following the above method. It is obvious that the routing path is non-monotonic if we insert buffer b_1 into tile c_{11} and buffer b_2 into tile c_{23}. Therefore, there is not an edge from c_{11} to c_{23} in the s-t graph.

As shown in Fig.2, the congestion of each grid is shown in the left-bottom corner. We can assign each edge a cost which is the congestion of the most congestive grid in the possible routing. For example, the cost of edge (c_{13}, c_{22}) is 3. By finding the single-pair shortest path, buffer b_1 is inserted tile c_{13}, and b_2 is inserted c_{21}.

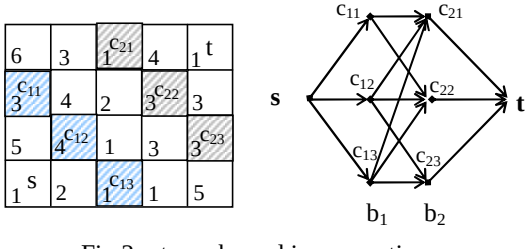

Fig.2 *s-t* graph used in congestion driven buffer insertion algorithm

D. Buffer Insertion with consideration of congestion

The overview of the congestion driven buffer insertion algorithm is shown in fig.3.

Steps 1 through 3 are data prepare stages. In the step 2, the set of candidate tile for each buffer is calculated. Without consideration of buffers we compute each net's contribution to the congestion in the step 3.

In step 4, for each net, the assignment from buffer to tile is achieved by solving a single-pair shortest-path problem, which is described in section IV.*B*. We will update the contribution of the net to the global congestion if the buffers of a net are all inserted successfully. Otherwise, the net is buffered unsuccessfully, which means there is no feasible dead space for buffer insertion, or all of the possible routing path constrained by buffers are non-monotonic.

Algorithm.1 Congestion Driven Buffer Insertion

1. Build the tile data structure for all the dead-spaces.
2. Compute the set of candidate tiles for each buffer.
3. Build grid data structure for congestion estimation and initialize the congestion value for each congestion grid.
4. For each net, perform the following operations.
   Construct the s-t graph G.
   Calculate cost for each edge.
   Find the shortest path, if the graph G is connected.
   According to the shortest path, the buffers are inserted.
   Update the congestion information.

Fig.3 Overview of the buffer insertion algorithm

V. CONGESTION OPTIMIZATION BY CHANGING THE DISTRIBUTION OF THE DEAD SPACE

In this section, we try to optimize the performance of the chip by redistributing the dead-spaces all over the placement. The above congestion-driven buffer planning algorithm is embedded into the optimization procedure. The objective is to maximize the number of nets that meet the delay constraints and minimize the congestion. The cost function is shown as follows.

$$Cost = p_1 * (TN - N) + p_2 * \lambda * C$$

In which the p_1 and p_2 is the weight for two different objectives, *TN* is the number of total two-pin nets, *N* is the number of nets that satisfy the delay constraint, λ is balance factor and *C* is the average of the congestion of the top 5% most congestive grids. Fig.4 outlines the Optimization algorithm.

Step 1 find all the dead space in the placement, and associate dead space with circuit block. The decomposition of the multi-pin nets follows. Steps 3 to 4 inserted some channels according to method discussed in section III.

Step 6 and step 8 perform the congestion-driven buffer insertion algorithm proposed in section IV. The congestion is evaluated by the average of the top 5% most congestive grids. Because of the timing consumption of the buffer insertion algorithm, steps 7 to 9 perform a local search on the distribution of the dead space.

In step 7, the new distribution of the dead space in the placement can be generated by the following methods.

One dead space block is randomly selected, and the associated block is moved to change the dead space distribution around the block. The dead space have been dissect into tiles, hence we move the block by random times of width or height of tile. And then we update the length and buffer number of the nets that have pins in the moved circuit block and independent feasible region of each buffer. If the new dead-space distribution is not accepted, we erase all the changes that have made.

Algorithm.2 Congestion Optimization

1. Compute all the dead-spaces in the placement, and associate each of the dead-spaces with some block.
2. Find the MST for each multi-pin net and decompose the multi-pin net into two-pin nets according to the edges of the MST.
3. Calculate the number of buffers needed for each net and IFR for each buffer and estimate the distribution of the buffers roughly.
4. Insert channels, update the dead-spaces in the placement.
5. Update the number of buffers needed for each net and calculate IFR for each buffer.
6. Perform the congestion driven buffer insertion algorithm to compute the number of nets that meet the target delay, denoted as N_{old} and the average of the congestion of the top 5% most congestive grids, denoted as C_{old}, $Cost_{old} = p_1*(TN-N_{old}) + p_2*\lambda*C_{old}$.
7. Generate new distribution of the dead-spaces and update related information.
8. Perform the congestion-driven buffer insertion algorithm to compute the number of nets that meet the delay constraints, denoted as N_{new}, and congestion C_{new}, $Cost_{new} = p_1*(TN - N_{new}) + p_2*\lambda*C_{new}$.
9. If $Cost_{old} < Cost_{new}$, $Cost_{new} = Cost_{old}$, $N_{old} = N_{new}$, $C_{old} = C_{new}$, the new dead space distribution is accepted. Otherwise, restore the changes.
10. Repeat steps 6 to 9 until no improvement in given times.

Fig.4 Overview of the optimization algorithm

The topology and total area of the placement keep unchanged during the dead-space redistribution. We perform the previous congestion driven buffer insertion algorithm to compute the number of nets that meet the target delay and the congestion cost under each new distribution of the dead space.

VI. EXPERIMENTAL RESULTS

The congestion driven buffer insertion algorithm and the congestion optimization algorithm have been implemented using C++ language on a Pentium III 733 machine. In this section, we present some details of our experimental results obtained. The values for parameters shown in Table.1 are based on a 0.18μm technology in the NTRS'97 roadmap [17]. In the implementation, the size of tiles used for buffer insertion is selected as four times of the buffer size and the congestion grid size is four times of tile size.

We ignore all power and ground interconnects. The initial placements of the MCNC benchmark circuits were obtained from [1]. We assign target delays of the two-pin nets, since the MCNC benchmarks include no any timing information. All two-pin nets whose lengths are smaller than the critical length $l_{\min}$ [6] are ignored, because buffer insertion cannot reduce their delay. And then we compute the optimal delay T_{opt} under optimal buffer insertion [6] for each net and then

randomly assign a constraint delay between 1.05 and 1.20 times T_{opt} to the net as in [2,5]. We provide the results of our algorithm for 5 MCNC benchmark circuits [7]. The details of these circuits are shown in Table.2.

TABLE 1 VALUE FOR THE PARAMETERS USED

r	Unit length wire resistance (Ω/μm)	0.075
c	Unit length wire capacitance (fF/μm)	0.118
T_b	Intrinsic buffer delay (ps)	36.4
C_b	Buffer input capacitance (fF)	23.4
R_b	Buffer output resistance (Ω)	180
R_d	Driver output resistance (Ω)	180
C_l	Sink input capacitance (fF)	23.4

TABLE 2
MCNC BENCHMARKS STATISTICS

Circuit	Blocks	Nets	Two-pin Nets
Apte	9	97	172
Xerox	10	203	455
Hp	11	83	226
Ami33	33	123	363
Ami49	49	408	545

In Table.3 and Table.4, we provide some experimental results from our buffer planner, which include running the congestion-driven buffer insertion algorithm in the original placement (BP1), in the placement with inserted channels (BP2) and in the optimization procedure (BP3). Each of the situations includes the number of nets which meet the delay constraint (#meet), the total number of buffers of nets that satisfy target delay and the total number of buffers needed (#IB/#TB), the area usage, the routing congestion and CPU times. In BP3 the area usage is not shown, because the total area is not changed during all the optimization stage. Table.3 also shows the comparison between BP1 and BP2 (BP2 to BP1) by the ratios of improvement. Table.4 shows the comparison between BP2 and BP3 (BP3 to BP2), and BP1 and BP3 (BP3 to BP1). All of the experimental results include nets that can not meet the timing constraints after channel insertion and nets that are shorter than the critical length l_{min} [6]. We enlarge all the circuit to involve more buffers to test our algorithms.

Since the method of decomposing the multi-pin nets is different from each other, it is meaningless to make a comparison with the results from [2] and [5]. However, we list the experimental results in [15] (BP [15]) for references.

Comparing BP3 with BP1, we can find that the performance of the chip is improved greatly on penalty of a small area usage. For circuit Xerox, the number of nets that meet the delay constraints increases 24.3% by a cost of 1.1% area usage and the congestion is decreased 17.2%.

The results in Table.4 also show that changing the distribution of the dead space is able to reduce routing congestion. With p_1=1 and p_2=0 the objective function is oriented towards the increasing of the nets satisfying the target delay, whereas setting p1=0 and p2=1 causes the routing congestion to be reduced. p_1=p_2=0.5 represents a solution with a tradeoff between the two different objectives. According to BP2 and BP3, changing the distribution of the dead space maybe improve the routing congestion, and simultaneously increase the number of delay-met nets. For example, in the circuit Xerox, the number of nets meeting target delay increases 5.2% and the routing congestion decrease 4.2%, and the area usage decreases 1.1%. However, the area usage of five test cases decreases 2.08% averagely and the optimization procedure is very time consuming.

Fig.5 (a), (b), and (c) illustrate the experimental results of the circuit Xerox under three different situations. The very small rectangles (more like points) are buffers that have been inserted. In Fig.5 (c), the upper-left block and upper-right block are moved so that the dead space distribution is changed, and more buffers are inserted. From Fig.5, we can observe the effectiveness of the proposed algorithms clearly.

VII. CONCLUSION

This paper proposed a congestion-driven buffer insertion algorithm, in which single-pair shortest-path model is used for computing the optimal buffer location and preserving monotonicity of routing paths. In order to get more buffers inserted, on the basis of a rough estimation of buffer location, some channels determined by the boundary of circuit block are inserted. Furthermore, we change the distribution of the dead space to perform an optimization for buffer insertion and routing congestion, which can be improved greatly by costing small area usage. The efficiency of proposed algorithm has been demonstrated by the experimental results.

Because of time consumption, a greedy local search algorithm is used for optimization in our paper. Though the experimental results show that the greedy strategy is efficient, it is required to develop a faster buffer planning algorithm for applying a better search strategy.

References

[1] X.L. Hong, G. Huang, Y.C. Ma, Yici Cai, S.Q. Dong, "Corner Block List: an effective and efficient topological representation of non-slicing floorplan," ICCAD'2000.

[2] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning", IEEE/ACM ICCAD, 1999.

[3] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies," Frontiers in Semiconductor Research: A collection of SRC Working Papers, Semiconductor Research Corporation, http://www.src.org/prg_mgmt/frontier.dgw, 1997

[4] J. Cong, L. He, C-K. Koh, P.H. Madden, "Performance optimization of VLSI interconnect layout" Integration, the VLSI Journal, vol.21, Nov. 1996.

[5] P. Sarkar, C. K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," International. Symposium on Physical Design, 2000.

[6] C.J. Alpert, A. Devgan, "Wire segmenting for improved buffer insertion," in Proc. Design Automation Conf, June 1997.

[7] Collaborative Benchmarking Laboratory, North Carolina State University, *http://www.cbl.ncsu.edu/CBL Docs/lys92.html: LayoutSynth'92 Benchmark Information*.

[8] X. Tang and D.F. Wong, "Planning buffer locations by network flows", Intl. Symp. Physical Design, 2000, pp. 180-185.

[9] F. F. Dragan, A. B. Kahng, I. Mandoiu, S. Muddu, "Provably good global buffering using an available buffer block plan", IEEE/ACM ICCAD, 2000

[10] F. F. Dragan, A. B. Kahng, et al"Provably good global buffering by multiterminal multicommodity flow approximation", ASP-DAC, 2001.

[11] C.J. Alpert, J. Hu, S.S. Sapatnekar, P.G. Villarrubia, "A practical methodology for early buffer and wire resource allocation," DAC, 2001.

[12] J. M. Ho and G. Vijayan and C. K. Wong, "New algorithms for the rectilinear Steiner tree problem", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, Volume: 9 Issue: 2, Feb. 1990.

[13] C. W. Sham, F. Y. Young, " Routability driven floorplanner with buffer block planning", ISPD 2002.
[14] F. Ragiq, M. C. Jeske, H. H. Yang, N. Sherwani, "Integrated floorplanning with buffer/channel insertion for bus-based microprocessor designs", ISPD 2002.
[15] S. Chen, X. L. Hong, S. Q. Dong, Y. C. Ma, Y. C. Cai, et al, "A buffer planning algorithm based on dead space redistribution", ASP-DAC 2003.
[16] J. Lou, S. Thakur, S. Krishnamoorthy, H.S. Sheng, "Estimating Routing Congestion Using Probabilistic Analysis", IEEE transaction on computer-aided design of integrated circuits and systems. vol. 21, no.1, January 2002.
[17] Semiconductor Industry Association, National Technology Roadmap for Semiconductors, 1997.
[18] Y. Ma, X. Hong, S. Dong, S. Chen, Y. Cai, et al, "Dynamic Glocal Buffer Planning Optimization Based on Detail Block Locating and Congestion Analysis", DAC 2003.
[19] K. W.C. Wong, F. Y. Yong, "Fast Buffer Planning and Congestion Optimization in Interconnect-driven Floorplanning", ASP-DAC 2003.

(a) Initial placement (b) Channels Inserted (c) Dead-space distribution optimized

Fig.5 Results of circuit Xerox

TABL.3 EXPERIMENTAL RESULTS OF CHANNEL INSERTION

circuit	BP1					BP2					BP2 to BP1		BP[15]	
	#meet	#IB/TB	usage(%)	congestion	time(s)	#meet	#IB/TB	usage(%)	congestion	time(s)	#meet	#usage	#meet	#IB
apte	78	15/156	96.9	18.18	1.65	116	71/159	95.9	16.89	1.52	48.70%	1.00%	100	104
xerox	313	57/155	93.9	23.51	2.35	366	125/162	92.8	20.07	2.29	16.90%	1.10%	315	182
hp	152	6/59	90.9	13.85	0.05	169	27/60	89.1	11.49	0.07	11.20%	1.80%	139	182
ami33	308	28/71	90.9	8.48	0.16	320	47/81	88.6	8.88	0.28	3.90%	2.30%	249	178
ami49	412	173/199	90.1	27.23	1.33	456	241/265	85.9	27.96	2.36	10.70%	4.20%	457	253

TABLE.4 EXPERIMENTAL RESULTS OF OPTIMIZATION

circuit	BP3					BP3 to BP2		BP3 to BP1		
	p1, p2	#meet	#IB/TB	congestion	time(s)	#meet	congestion	#meet	#congestion	#usage
apte	p1=1, p2=0	117	69/159	16.89	99.12	0.09%	0.00%	50.00%	7.10%	1.00%
	p1=p2=0.5	117	69/159	16.89	98.90	0.09%	0.00%	50.00%	7.10%	
	p1=0, p2=1	117	69/159	16.89	98.96	0.09%	0.00%	50.00%	7.10%	
xerox	p1=1, p2=0	389	154/174	19.69	317.70	6.30%	2.00%	24.30%	16.20%	1.10%
	p1=p2=0.5	389	152/174	19.46	655.67	6.30%	3.40%	24.30%	17.20%	
	p1=0, p2=1	385	149/177	19.23	620.44	5.20%	4.20%	23.00%	18.20%	
hp	p1=1, p2=0	193	48/61	12.10	63.91	14.20%	-5.00%	27.00%	12.60%	1.80%
	p1=p2=0.5	190	47/61	10.52	92.37	12.40%	8.40%	25.00%	24.00%	
	p1=0, p2=1	174	43/66	9.89	92.37	3.00%	13.93%	14.50%	28.60%	
ami33	p1=1, p2=0	335	69/95	8.13	447.34	4.90%	8.45%	8.80%	4.10%	2.30%
	p1=p2=0.5	335	69/95	8.13	452.50	4.90%	8.45%	8.80%	4.10%	
	p1=0, p2=1	330	63/93	8.04	374.46	3.10%	8.00%	7.10%	7.53%	
ami49	p1=1, p2=0	472	255/262	28.35	471.87	3.50%	-1.37%	14.60%	-4.00%	4.20%
	p1=p2=0.5	461	238/264	27.50	212.80	0.44%	1.60%	11.90%	-1.00%	
	p1=0, p2=1	461	238/264	27.50	210.17	0.44%	1.60%	11.90%	-1.00%	