Buffer Block Planning for Interconnect Planning and Prediction

Jason Cong, *Fellow, IEEE*, Tianming Kong, *Student Member, IEEE*, and Zhigang (David) Pan, *Member, IEEE*

***Abstract*—This paper studies buffer block planning (BBP) for interconnect planning and prediction in deep submicron designs. We first introduce the concept of feasible region for buffer insertion, and derive its closed-form formula. We observe that the feasible region for a buffer is quite large in general even under fairly tight delay constraint. Therefore, it gives us a lot of flexibility to plan for buffer locations. We then develop an effective BBP algorithm to perform buffer clustering such that design objectives such as overall chip area and the number of buffer blocks can be minimized. Effective BBP can plan and predict system-level interconnect by construction, so that accurate interconnect information can be used in early design stages to ensure design closure.**

***Index Terms*—Buffer insertion and planning, feasible region, floorplanning, interconnect planning and prediction.**

I. Introduction

FOR deep submicron (DSM) very large scale integrated (VLSI) designs, it is well known that interconnect has become the dominant factor in determining the overall circuit performance and complexity. To improve the interconnect performance, many interconnect optimization techniques have been proposed recently such as topology construction, driver sizing, buffer insertion, wire sizing, and spacing (see [1] and [2] for a tutorial). Among them, buffer insertion, in particular, is a very effective and useful technique by inserting active devices (buffers) to break original long interconnects into shorter ones so that the overall delay can be reduced. It has been shown that without buffer insertion, the interconnect delay for a wire increases about quadratically as wire length increases, but it only increases linearly under proper buffer insertion [3]–[6]. As an example, it was shown in [2] that the delay of a 2-cm global interconnect can be reduced in a factor of 7× by the optimal buffer insertion. As the intrinsic delay of a buffer becomes smaller and the chip dimension gets larger, it is expected that more and more buffers shall be inserted for high-performance integrated circuit (IC) designs (e.g., close to 800 000 for the 50-nm technology as estimated in [7] and [8]). The introduction of so many buffers will significantly change a floorplan and wire length distribution. Thus they shall be planned as early as possible. By embedding such a buffer planning process into physical design flow (e.g., during floorplanning), one can estimate the interconnect parameters (length, timing, and so on) accurately for each individual net so that timing closure and design convergence may be better achieved.

However, most existing buffer insertion algorithms (e.g., [9]–[12]) were designed for post-layout interconnect optimization and for a single net only. There was no global planning for tens of thousands of nets that may need buffer insertion to meet their performance requirement as in DSM designs. Meanwhile, most existing floorplanning algorithms (e.g., [13]–[15]) only focused on wirelength/area minimization and did not consider buffer insertion for performance optimization. In [16], buffer insertion was considered during floorplanning, but it simply assumed that buffers can be inserted anywhere in an existing floorplan, which is not a realistic assumption since buffers must consume silicon resources and require connections to the power/ground networks [e.g., they cannot be inserted inside some hard intellectual property (IP) blocks]. Otherwise, it will seriously affect the hierarchical design style and make it difficult to use/reuse IP blocks. As a result, the designers often prefer to form buffer blocks between existing circuit blocks of the current floorplan. If there is no careful planning of these large amount of buffers, one may get excessive area increase. Moreover, without careful planning, it is most likely that these buffers will be distributed rather randomly over the entire chip, which will definitely complicate global/detailed routing and power/ground distribution.

To effectively address the above issues, as part of our general effort of developing an interconnect-centric design flow [17], [8], we study in this paper the buffer block planning (BBP) problem, which automatically generates buffer blocks for interconnect optimization during physical-level floorplanning. It considers buffer location constraints (e.g., hard IP blocks and predesign layout), and provides more regular buffering structure for layout and power/ground networks. Since the BBP roughly determines every buffer location for each net, we can then obtain more accurate wire length and congestion estimation and prediction during physical design.

Our major contributions of this paper includes the following.

- We first introduce the concept of feasible region (FR) for buffer insertion under certain delay constraint and derive an analytical formula for it.
- We find that the FR for a buffer can be surprisingly large, even under tight delay constraints. This crucial observation provides us a lot of flexibility to plan a buffer's location.
- We propose to use FRs to cluster individual buffers together to form buffer blocks so that the total chip area due

Manuscript received October 14, 2000; revised March 5, 2001. This work was supported in part by the Semiconductor Research Corporation under Contract 98-DJ-605 and a Grant and Equipment donation from Intel.

J. Cong and T. Kong are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: cong@cs.ucla.edu; kongtm@cs.ucla.edu).

Z. Pan is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: dpan@watson.ibm.com).

Publisher Item Identifier S 1063-8210(01)07660-0.

TABLE I
KEY PARAMETERS

r	unit length wire resistance (Ω/μm)	0.075
c	unit length wire capacitance (fF/μm)	0.118
T_b	intrinsic delay for buffer (ps)	36.4
C_b	input capacitance of buffer (fF)	23.4
R_b	output resistance of buffer (Ω)	180

to buffer insertion, as well as the number of buffer blocks, can be minimized.

- We develop an effective algorithm for BBP. It can be used as a key element for interconnect-driven floorplanning, interconnect planning, and interconnect estimation.

To the best of our knowledge, this is the first in-depth study of BBP. The rest of the paper is organized as follows. Section II formulates the problem. Section III derives the feasible region for buffer insertion. Section IV studies BBP and proposes an effective algorithm for it. Experimental results are shown in Section V, followed by the conclusion in Section VI. Part of the preliminary results of this work were presented in [18].

II. Problem Formulation

The problem of BBP is formulated as follows. Given an initial floorplan and the performance constraints for each net, we want to determine the optimal locations and dimensions of buffer blocks (i.e., the extra blocks between existing circuit blocks of the current floorplan) such that the overall chip area and the number of buffer blocks after buffer insertion are minimized while the performance constraint for each net is satisfied (if it is a valid timing constraint that can be met by optimal buffer insertion under the given floorplan). The output from our BBP consists of the following information: the number of buffer blocks, each buffer block's area, location, and corresponding nets that use some buffer in this buffer block to meet the delay constraints. In this study, we focus on two-pin nets and derive the closed-form formula of feasible region for buffer insertion. The concepts of feasible region and BBP can be extended to multiple-pin nets as well.

The key parameters for interconnect and buffer in our study are listed in Table I. The values are based on a 0.18 μm technology in NTRS'97 [19]. In the table, the unit interconnect resistance and capacitance values are obtained based on a wire width of 0.9 μm and a wire spacing of 1.2 μm. The capacitance value is extracted using the three-dimensional (3-D) field-solver Fastcap [20]. We model a driver/buffer as a switch-level RC circuit [2], and use the well-known Elmore delay model for delay computation. The buffer consists of two inverters. They are 20 and 100 $\times$ the minimum transistor size, respectively, so that they can drive a fairly long interconnect. The buffer parameters are obtained using HSPICE simulations.

III. Reasible Region for Buffer Insertion

The FR for a buffer is defined to be the maximum region where the buffer may be located such that by inserting the buffer into any location in that region, the delay constraint can be satisfied. Fig. 1 illustrates the concept of FR for inserting 1 or k buffers into a net where the source and sink of the net are connected by a given route. In the figure, the FRs are the shaded line segments.

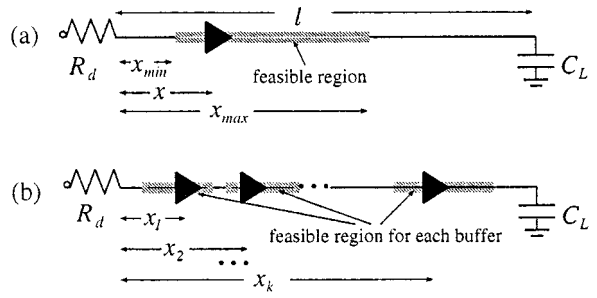

Fig. 1. Feasible regions for inserting (a) one buffer and (b) k buffers.

A. Feasible Region for Single-Buffer Insertion

For single-buffer insertion in Fig. 1(a), let us denote x to be the length from driver to buffer. We have the following theorem for its feasible region.

Theorem 1: For a given delay constraint $T_{\rm req}$, the feasible region $[x_{\min}, x_{\max}]$ for inserting one buffer is

$$x_{\min} = MAX\left(0, \frac{K_2 - \sqrt{K_2^2 - 4K_1K_3}}{2K_1}\right)$$
$$x_{\max} = MIN\left(l, \frac{K_2 + \sqrt{K_2^2 - 4K_1K_3}}{2K_1}\right)$$

where

K_1 rc;
K_2 $(R_b - R_d)c + r(C_L - C_b) + rcl$;
K_3 $R_dC_b + T_b + R_b(C_L + cl) + (1/2)rcl^2 + rlC_L - T_{\rm req}$.

Proof: For a single buffer insertion at length x from the driver, the Elmore delay from the driver to the sink is

$$\begin{aligned} T &= R_d(cx + C_b) + \tfrac{1}{2}rcx^2 + rxC_b + T_b \\ &\quad + R_b[c(l-x) + C_L] + \tfrac{1}{2}rc(l-x)^2 + r(l-x)C_L \\ &= rc \cdot x^2 - [(R_b - R_d)c + r(C_L - C_b) + rcl] \cdot x \\ &\quad + R_dC_b + T_b + R_b(C_L + cl) + \tfrac{1}{2}rcl^2 + rlC_L \\ &\le T_{\rm req}. \end{aligned}$$

Solving the above quadratic inequality with the boundary condition that the buffer has to be placed between driver and receiver, we get $x_{\min}$ and $x_{\max}$ as stated in the theorem. □

Note that for Theorem 1 to be valid, $K_2^2 - 4K_1K_3 \ge 0$ shall hold. Otherwise, no feasible region exists and the initial floorplanning/timing budget has to be modified. Fig. 2 shows the feasible region for inserting one buffer to an interconnect of length from 6 to 9 mm. We first compute the best delay $T_{\rm best}$ by inserting one buffer, then assign the delay constraint to be $(1+\delta)T_{\rm best}$, with δ to be from 0 to 50%. The x axis shows the δ and the y axis shows the FR length, i.e., $x_{\max} - x_{\min}$. It is interesting to see that even with a fairly small amount of slack, say 10% more delay from $T_{\rm best}$, the FR can be as much as 50% of the wire length. This important observation leads to great flexibility for buffer planning, to be discussed later on in this paper.

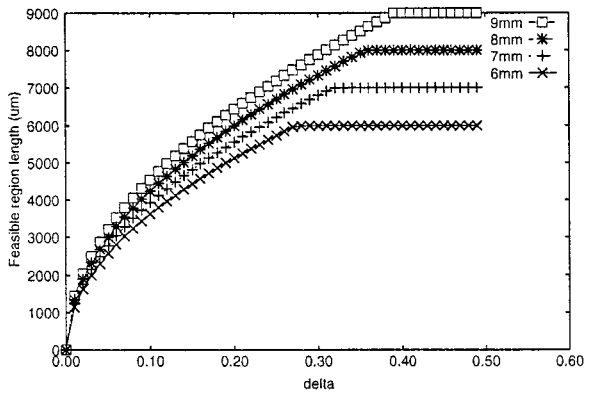

Fig. 2. The feasible region length for inserting one buffer under different delay constraints of δ.

B. Feasible Regions for Multiple-Buffer Insertion

For a long interconnect, more than one buffer may be needed to meet a given delay budget. For k buffers inserted, we have the following theorem to compute the feasible region for each buffer.

Theorem 2: For a long interconnect with k buffers inserted, the feasible region for the ith buffer ($i \leq k$) is $x_i \in [x_{\min}(k, i), x_{\max}(k, i)]$ with

$$x_{\min}(k,i) = MAX\left(0, \frac{K_2' - \sqrt{K_2'^2 - 4K_1'K_3'}}{2K_1'}\right)$$
$$x_{\max}(k,i) = MIN\left(l, \frac{K_2' + \sqrt{K_2'^2 - 4K_1'K_3'}}{2K_1'}\right)$$

where K_1', K_2', and K_3' are

$$K_1' = \frac{(k+1)rc}{2i(k-i+1)}$$
$$K_2' = \frac{(R_b - R_d)c}{i} + \frac{r(C_L - C_b) + rcl}{k-i+1}$$
$$K_3' = kT_b - T_{\text{req}} + \left[R_d + (k-1)R_b + \frac{(k-i)rl}{k-i+1}\right]\cdot C_b + R_b(C_L + cl) + \frac{rcl^2}{2(k-i+1)} + \frac{rlC_L}{k-i+1} - \frac{(i-1)c(R_b - R_d)^2}{2ir} - \frac{(k-i)r(C_b - C_L)^2}{2(k-i+1)c}.$$

Proof: The proof is similar to that of Theorem 1. The main difference is that with total number of k buffers, for the ith buffer, there are $i-1$ buffers before it and $k-i$ buffers after it. Since we want to compute the maximum region that the ith buffer can be placed, all other buffers are assumed to be optimally placed to minimize the delay during the computation of the feasible region of the ith buffer.

From [21, eq. (6)][1] we can obtain the optimal delay for inserting j buffers into a wire of length l with driver resistance R_d and loading capacitance C_L

$$T_j(R_d, C_L, l) = \frac{rl(jC_b + C_L) + cl(R_d + jR_b)}{j+1} + \frac{(jC_b + C_L)(jR_b + R_d)}{j+1} + jT_b + \frac{rcl^2 - \frac{jr(C_b - C_L)^2}{c} - \frac{jc(R_b - R_d)^2}{r}}{2(j+1)}.$$

Then, for the ith buffer at x_i from the driver, we have the delay

$$\begin{aligned} T &= T_{i-1}(R_d, C_b, x_i) + T_b + T_{k-i}(R_b, C_L, l - x_i) \\ &= \frac{rx_i iC_b + cx_i[R_d + (i-1)R_b]}{i} + \frac{iC_b[(i-1)R_b + R_d]}{i-1+1} + (i-1)T_b + \frac{rcx_i^2 - \frac{(i-1)c(R_b - R_d)^2}{r}}{2(i)} + T_b \\ &\quad + \frac{r(l-x_i)[(k-i)C_b + C_L] + c(l-x_i)(k-i+1)R_b}{k-i+1} + \frac{[(k-i)C_b + C_L](k-i+1)R_b}{k-i+1} \\ &\quad + (k-i)T_b + \frac{rc(l-x_i)^2 - \frac{(k-i)r(C_b - C_L)^2}{c}}{2(k-i+1)} \\ &= K_1' \cdot x_i^2 - K_2' \cdot x_i + (K_3' + T_{\text{req}}) \\ &\leq T_{\text{req}}. \end{aligned}$$

Solving the above inequality, we get $x_{\min}(k, i)$ and $x_{\max}(k, i)$ as stated in the theorem. □

It can be verified that Theorem 1 is a special case of Theorem 2 with $k = i = 1$. The following theorem determines the minimum number of buffers that are required to meet a given delay budget.

Theorem 3: The minimum number of buffers to meet the delay constraint T_{req} for an interconnect of length l is

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

where

$$K_4 = R_bC_b + T_b \tag{2}$$
$$K_5 = T_{\text{req}} + \frac{r}{c}(C_b - C_L)^2 + \frac{c}{r}(R_b - R_d)^2 - (rC_b + cR_b)l - T_b - R_dC_b - R_bC_L \tag{3}$$
$$K_6 = \tfrac{1}{2}rcl^2 + (rC_L + cR_d)l - T_{\text{req}}. \tag{4}$$

Proof: From [21, eq. (6)], we know that the optimal delay for inserting k buffers into a wire of length l is

$$T_k = \frac{rl(kC_b + C_L) + cl(R_d + kR_b)}{k+1} + \frac{(kC_b + C_L)(kR_b + R_d)}{k+1} + kT_b + \frac{rcl^2 - \frac{kr(C_b - C_L)^2}{c} - \frac{kc(R_b - R_d)^2}{r}}{2(k+1)}.$$

[1]There is a typo in [21, eq. (6)], which we have corrected.

Let $T_k < T_{\text{req}}$, and we can obtain the quadratic inequality of k in the form of $K_4 \cdot k^2 - K_5 \cdot k + K_6 < 0$. Solving the inequality, we prove Theorem 3. □

Based on these results, given a two-pin net with a delay constraint T_{req}, the required buffer number $k_{\min}$ and the feasible region for each buffer can be computed in constant time. As a example, for a 1-cm net with $R_d = R_b$, $C_L = C_b$ and the delay constraint $T_{\text{req}} = 1.05 \cdot T_{\text{best}}$ (T_{best} is the best delay by optimal buffer insertion, which is 464 ps), we can calculate that the minimum number of buffers needed is $k_{\min} = 2$, and the feasible regions for the first and second buffers are [1.47 mm 5.20 mm] and [4.80 mm 8.53 mm], respectively. Note that the FRs of adjacent buffers may overlap, as in this example. This is because FR for each buffer is computed independently, assuming all other buffers can be optimally placed to satisfy the delay constraint, i.e., our FR provides maximum freedom for each buffer. It shall be noticed that during the buffer planning phase (in Section IV), when a buffer is placed (i.e., "committed") to a position within its feasible region, we will need to update the FRs of all other unplaced buffers of the same net to safely meet its delay constraint. But since we have the analytical formula, this update can be computed in constant time.

C. Two-Dimensional (2-D) Feasible Region

So far, our discussion of FR is restricted to a one-dimensional (1-D) line, i.e., we assume the route from source to sink is already specified by some global router. Thus the feasible region is also 1-D. In practice, however, global routing usually has not been performed prior to or during floorplanning. In this case, we can compute a much larger two-dimensional (2-D) FR for each buffer. This 2-D feasible region is essentially the union of the 1-D feasible regions of all possible routes from source to sink. Therefore, we can have much more freedom for buffer planning. Since for each net, its buffer location will then determine roughly its routing, our BBP indeed determines the overall global routing structure for each net.

For a 2-D net, let the source location be (x_{src}, y_{src}) and the sink location be $(x_{\text{sink}}, y_{\text{sink}})$. We only need to consider nondegenerate 2-D cases here, i.e., $x_{src} \neq x_{\text{sink}}$ and $y_{src} \neq y_{\text{sink}}$. Also, we consider only the monotone (i.e., nondetour) routes from source to sink. We prove that with Manhattan monotone routing, the 2-D FR can be obtained by the following theorem.

Theorem 4: For a net with k buffers, the 2-D feasible region for the ith buffer is the region bounded by the rectangular bounding box between the source and the sink and by two parallel lines with Manhattan distances from the source to be $x_{\min}(k, i)$ and $x_{\max}(k, i)$, respectively (the same as Theorem 2). The slope of the two parallel lines is either $+1$ or -1, depending on the sign of $(y_{\text{sink}} - y_{src})/(x_{\text{sink}} - x_{src})$: if $(y_{\text{sink}} - y_{src})/(x_{\text{sink}} - x_{src}) > 0$, the slope is -1; if $(y_{\text{sink}} - y_{src})/(x_{\text{sink}} - x_{src}) < 0$, the slope is $+1$.

Proof: It can be easily verified from the definition of Manhattan distance and monotone routing. □

Note that in previous works of buffer insertion, buffers are mostly inserted in their delay-minimal positions, which we call restricted positions because they are only a small subset within our FR. The restricted positions for a 2-D net can be obtained by the following corollary.

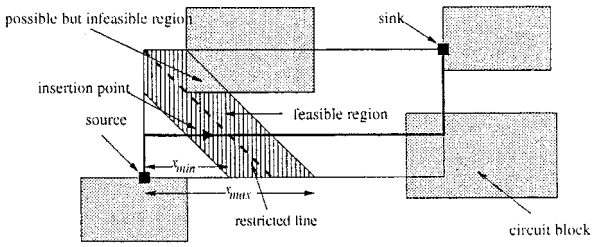

Fig. 3. The 2-D feasible region and a restricted line. The existing circuit blocks act as obstacles for buffer insertion.

Corollary 1: For a 2-D net with k buffers, the restricted positions of the ith buffer for all monotone routes from the source to the sink form a restricted line within the feasible region of the ith buffer. The line slope is again either $+1$ or -1, the same as that in Theorem 4. □

Also, if there are obstacles (such as hard IP blocks), we just need to deduct them from the feasible region. An example of a 2-D feasible region with a restricted line and some obstacles is illustrated in Fig. 3.

IV. Buffer Block Planning

In the previous section, we show that for a given delay constraint, a buffer may be inserted in a fairly wide feasible region. Therefore, it gives us a lot of flexibility to plan for every buffer's insertion position (within its FR) such that the overall chip area due to buffer insertion, as well as the total number of buffer blocks can be minimized. It shall be noted that such a BBP also determines the overall global routing structures for long interconnects by determining their internal buffer locations.

The BBP problem is very difficult in the following senses: 1) many buffer blocks might need to be optimally shaped for overall chip area minimization and 2) to make the situation even more complicated, different buffers of the same net will not be independent of each other. For a long interconnect with more than one buffer inserted, Theorem 2 gives the maximum FR for each buffer. However, when a buffer is committed to a certain location within its FR, the FRs for other buffers in the same net will have to be updated so that the delay constraint can be safely met.[2]

In the rest of this section, we will present an effective algorithm to solve the BBP problem. There are several important features in our BBP algorithm: 1) it takes advantage of both the flexibility of FR and the simplicity of its analytical formulas so that one may handle large circuits with tens of thousands long interconnects easily; 2) since in most floorplans there are some dead areas that cannot be taken by any circuit module, our algorithm will use these dead areas as much as possible to save the overall chip area; and 3) different from the previous buffer insertion algorithm that only inserts one buffer for a single net, our BBP algorithm always maintains global buffer insertion information for all nets, thus, it can effectively cluster individual buffers that belong to different nets into buffer blocks.

[2]Fortunately, we have the analytical formula to compute FR. Thus, the update can be done extremely fast.

```
Algorithm: Buffer Block Planning (BBP)
1. build horizontal and vertical polar graph;
2. build tile data structure;
3. for each tile, compute its area slacks;
4. while (there exists buffer to be inserted) {
5.       tile ← Pick_A_Tile();
6.       Insert_Buffers(tile);
7.       update W_c,H_c, FR and area slacks;
8. }
```

Fig. 4. Overall flow of the BBP algorithm.

Fig. 4 gives the overall flow of the BBP algorithm. Lines 1–3 are the data preparation stages. First, we will build the horizontal and vertical polar graphs [22], for the given floorplan denoted as G_H and G_V, respectively. Let us take G_H to illustrate how to build the horizontal polar graph. G_H is a directed graph, each vertex v in it corresponds to a vertical channel, and an edge $e = (v_1, v_2)$ corresponds to a circuit module whose left and right boundaries are adjacent to channels v_1 and v_2, respectively. For each vertex v, we assign its weight $w(v)$ to be its corresponding channel width. Similarly, for each edge e, we assign its weight $w(e)$ to be its corresponding module width. Graph G_V can be built similarly. By running the longest path algorithm on G_H/G_V, we can obtain the width/height of the chip (denoted by W_c/H_c). For those channels not on the critical paths in G_H/G_V, we will have some positive slacks in width/height, which lead to dead areas. It shall be noted that during buffer insertion, some circuit modules may have to shift to make room for buffer blocks (e.g., if no dead area exists). Therefore, the height of a horizontal channel or the width of a vertical channel may increase during BBP.

To better represent buffer block and facilitate data manipulation such as feasible region intersection, we divide each channel into a set of rectangular tiles. Then we compute for each tile τ, its slack with respect to the longest path in the polar graph G_H or G_V.

The **while** loop from lines 4 to 8 is the main part of our BBP algorithm. The iterative buffer insertion process will continue as long as there is still some net that needs buffer(s) to meet its performance constraint. Each iteration of the **while** loop has two major steps: first, we will pick a best tile for buffer insertion (**Pick_A_Tile**); then, we will insert proper buffers into this tile (**Insert_Buffers**).

To pick the best tile in each iteration, the **Pick_A_Tile** routine works in the following two modes, depending on whether there exists some useful dead area for buffer insertion or not.

1) There exists some tile whose area slack is positive (due to dead area). In this case, buffers inserted into this tile will not increase the overall chip area as long as the total area of buffers inserted in the tile is smaller than the area slack of this tile. For example of a tile τ in a vertical channel, suppose its width slack is w_τ, and its height is H_τ. Then we can insert as many as $\lfloor w_\tau H_\tau / A_b \rfloor$ buffers into tile τ without increasing the overall chip area, where A_b is the area of a buffer. The actual number of buffers that can be inserted into τ may be smaller, since only those buffers whose FR intersects with tile τ can be candidates to be inserted into τ. Therefore, the number of buffers that can be inserted into τ, without chip area increase is $n_\tau = \min(\lfloor w_\tau H_\tau / A_b \rfloor, m_\tau)$, where m_τ is the number of buffers whose FRs intersect with tile τ. Since we may have multiple tiles with positive slack (especially at the beginning of BBP), we will pick the one with largest n_τ because this strategy shall reduce the total number of buffer blocks, which is also our BBP objective.
2) There is no tile with positive area slack. Then, any buffer insertion will increase the overall chip area. When some buffer is inserted into a tile, we have to shift some circuit modules. This shifting will make room for other tiles, so we will have some new positive-slack tiles. Our tile selection process will try to maximize such opportunity. Notice that after a buffer is inserted in τ, other tiles in the same channel with τ will have positive area and tend to have buffers inserted in the future, thus the chance of buffers clustering increases. To maximize such effect, we will pick the channel that has the maximum buffer insertion demand and choose one tile in it. Note that in this scenario, since we need to expand the channel, we only insert one buffer into it to minimize the area increase.

Our strategy for **Insert_Buffers** into the tile τ that has just been picked by **Pick_A_Tile** also works in two modes, corresponding to those two in **Pick_A_Tile**:

1) The tile τ has dead area. From case 1 in **Pick_A_Tile**, we know that n_τ buffers can be inserted into the tile. Meanwhile, there are m_τ buffer candidates whose FRs intersect with tile τ, with $m_\tau \geq n_\tau$. Then if $n_\tau = m_\tau$, we will insert all these m_τ buffers; if $m_\tau > n_\tau$, we will only insert first n_τ buffers out of these m_τ buffers, sorted according to the increasing size of their FRs. Different from previous approaches that just inserts one buffer for one net, our approach inserts as many as n_τ buffers for n_τ different nets simultaneously. Since all of them are clustered into tile τ, they form a natural buffer block.
2) The tile does not have dead area, but needs expansion to make room for any buffer insertion. In this case, we only insert one buffer, i.e., $n_\tau = 1$. Again, if there are multiple buffers that can be inserted in this tile, we insert the one with the tightest FR constraint.

After deciding how many and which n_τ buffers are inserted ("committed") into tile τ, we will update the following information: 1) the feasible regions of "uncommitted" buffers in the same net for which we just inserted a buffer into τ; 2) the corresponding vertex (i.e., channel) weights in G_H and/or G_V that are affected by the insertion of the buffer block; and 3) the new chip dimension W_c, H_c and slacks for each channel and tile. Then we repeat the buffer insertion/clustering process until all buffers are placed. It shall be pointed that our BBP algorithm can handle both slicing and nonslicing floorplanning structures.

Note that during the above discussion, it is assumed that no area is pre-allocated for buffers, and we can adjust block locations (by expanding channels) as we insert buffers. However changing a floorplan may not always be feasible or desirable. For example, the change of floorplan may change datapaths, designer requirements, and so on. To address this issue, one would

TABLE II
TEST CIRCUIT STATISTICS

circuit	#modules	# nets	# pads	#2-pin nets
apte	9	97	73	172
xerox	10	203	2	455
hp	11	83	45	226
ami33	33	123	43	363
ami49	49	408	22	545
playout	62	2506	192	2150
ac3	27	212	75	446
xc5	50	1005	2	2275
hc7	77	449	51	1450
a9c3	147	1202	22	1613
pc2	124	3126	192	4204

use the following flow: assume the input floorplan has enough room reserved for buffer blocks, run our BBP algorithm to find out: 1) whether more spreading should be done; 2) if so, by how much; and 3) how, otherwise, what the buffer planning solution is. To fit into such applications, our algorithm needs some modification: at each iteration after we insert buffers, do not update the corresponding vertex weights in G_H and/or G_V or the new chip dimension as they are prefixed. Instead, we only need to update feasible regions of affected buffers, and for each tile how much room is used. In this case, our algorithm is slightly simpler. However, the floorplanner has to be able to consider buffer insertion and, thus, reserve room for buffer blocks. After BBP, a 2-D compaction may be applied to remove the excessive area reserved for buffer blocks.[3]

V. EXPERIMENTAL RESULTS

We have implemented our BBP algorithms using C++ on a 500-MHz Intel Pentium-III machine with 128 M-byte main memory. This section presents the experimental results. The parameters (refer to Table I) used in our experiments are based on a 0.18-μm technology in the NTRS'97 roadmap [19].

We have tested our algorithms on 11 circuits, as summarized in Table II. The first six circuits are from MCNC benchmark [23], and the other five are randomly generated. In this paper, we focus on 2-pin nets, so we decompose each multipin net into a set of source-to-sink 2-pin nets.[4] We then compute the critical length for buffer insertion (defined to be the minimal interconnect length that buffer insertion is needed for delay reduction) using the analytical formula in [21]. We then use it to filter out short interconnects, i.e., if a net is shorter than the critical length, we will ignore it during BBP since buffer insertion will not help reduce its delay. The initial floorplan for each circuit is generated by running the simulated tempering (an improved Monte Carlo technique of simulated annealing) algorithm as in [24]. For each net, we first compute its best delay by optimal buffer insertion T_{opt} [21], and then randomly assign its delay budget to be $1.05 \sim 1.20T_{\text{opt}}$.

We compare our BBP algorithm with a conventional buffer insertion algorithm without trying to plan buffer positions, i.e., at each iteration, a buffer is randomly picked and assigned to a feasible location, denoted as an random (RDM) algorithm. We run BBP and RDM under two different scenarios: one is labeled restricted (RES), where a buffer can only be located in its delay-minimal restricted position(s) (see Fig. 3) and the other is labeled FR where a buffer may be inserted anywhere in its feasible region. The results for four different algorithmic combinations are summarized in Table III, where BBP/RES means BBP algorithm applied to scenario RES, RDM/FR means RDM applied to scenario FR, and so on. The results are summarized in Table III.

It is interesting to observe from the table the following.

- Under the same algorithm, e.g., BBP, the usage of FR significantly increases the number of nets that can meet their delay constraints (for example of *ac3*, from 300 to 366, a 22% increase). This is because our feasible region is usually much larger than the delay-minimal RES locations, so that one can avoid existing circuit/buffer blockages during buffer insertion. Note that as *#meet* increases, the number of buffers inserted to meet performance constraints also increases accordingly from RES to FR. However, since the FR provides much more freedom during buffer clustering, the number of buffer blocks (*#BB*) in fact reduces (for example of of *a9c3*, from 542 to 365, a 33% reduction); and the area expansion due to buffer insertion is also less by using FR with better buffer clustering.
- Under the same FR or RES scenario, BBP algorithm can achieve much better area utilization than RDM. For example of *a9c3* and *pc2*, BBP/FR can achieve area ratio 23.90% and 26.49%, while RDM/FR can only achieve 5.60% and 5.89%, respectively. This is because BBP tries to cluster buffers for the overall area minimization.

  Note that for some circuit, even BBP/FR may not be able to achieve a very high area ratio. For example of *xerox*, BBP/FR only achieves a ratio of 4.32%. The reason is that the buffers of many nets cannot be inserted into tiles with positive area slack, thus their insertion will lead to overall area expansion. As an example, Fig. 5 shows the circuit buffer block layouts from RDM/RES and BBP/FR on circuit *xerox*. From the figure, we can see that BBP/FR indeed inserts many buffers into tiles which lie on the critical path of the horizontal polar graph. Such a problem is caused by the input floorplan which does not consider buffer planning. Using our BBP, however, can help to get a better interconnect-centric floorplanning.
- Under the same RES scenario (i.e., only the restricted positions are allowed for buffer insertion), the RDM and BBP algorithms will have about the same number of buffers inserted and the same number of nets meeting their delay constraints.[5] However, our BBP algorithm is able to explicitly cluster appropriate buffers together, so that it leads to significant area saving and much fewer number of buffer blocks than RDM algorithm. For

[3]We would like to thank the anonymous reviewers for pointing out this flow to us.

[4]Note that the number of 2-pin nets is possibly smaller than that of original nets (*playout*) because the power/ground and single-pin nets are excluded.

[5]A net fails to meet its delay constraint if the given delay constraint is too tight, or its buffer's feasible region is fully occupied by existing circuit blocks.

TABLE III
COMPARISON OF FOUR DIFFERENT BUFFER INSERTION/PLANNING ALGORITHMS. WE COMPARE THE TOTAL NUMBER OF BUFFERS INSERTED TO MEET PERFORMANCE CONSTRAINTS (*#buff*), THE NUMBER OF BUFFER BLOCKS (*#BB*), THE NUMBER OF 2-PIN NETS THAT CAN MEET THEIR DELAY CONSTRAINTS (*#meet*), THE CHIP AREA INCREASE DUE TO BUFFER INSERTION IN PERCENTAGE (*area*), AREA RATIO, I.E., THE RATIO OF TOTAL AREA OF ALL BUFFERS INSERTED AND TOTAL CHIP AREA INCREASE (*aratio*), AND THE TOTAL CPU TIME IN SECONDS (*cpu*)

	#buff	*#BB*	*#meet*	*area*	*aratio*	*cpu*	*#buff*	*#BB*	*#meet*	*area*	*aratio*	*cpu*
circuit	RDM/RES						RDM/FR					
apte	226	116	102	4.50%	14.46%	0.03	252	128	123	4.59%	15.81%	0.06
xerox	451	115	237	2.01%	2.60%	0.07	503	131	290	2.41%	2.41%	0.09
hp	283	154	138	3.20%	1.88%	0.07	294	176	148	3.35%	1.86%	0.08
ami33	641	271	257	2.96%	1.91%	0.21	703	294	307	3.06%	2.03%	0.31
ami49	850	389	322	3.01%	1.86%	0.45	941	429	392	3.27%	1.90%	0.60
playout	4002	614	1324	4.07%	5.13%	2.41	4343	741	1565	4.03%	5.62%	3.01
ac3	610	272	288	2.73%	4.07%	0.18	711	297	358	2.88%	4.50%	0.26
xc5	2596	448	1336	4.23%	6.32%	1.24	2941	477	1665	4.35%	6.95%	1.69
hc7	2672	905	946	6.40%	1.78%	2.15	2817	932	1072	6.60%	1.82%	2.93
a9c3	4032	1090	1199	4.35%	5.39%	4.84	4280	1170	1424	4.45%	5.60%	5.99
pc2	12166	1243	2815	8.52%	5.29%	14.15	12913	1312	3321	8.12%	5.89%	18.94
	BBP/RES						BBP/FR					
apte	225	76	98	2.67%	24.27%	0.04	262	56	132	1.44%	52.38%	0.06
xerox	442	78	222	1.37%	3.73%	0.08	519	61	304	1.39%	4.32%	0.10
hp	283	85	135	1.52%	3.96%	0.05	301	60	154	1.05%	6.10%	0.08
ami33	630	155	240	1.21%	4.60%	0.17	703	110	302	0.93%	6.70%	0.22
ami49	874	191	338	1.06%	5.44%	0.34	949	129	398	0.65%	9.58%	0.46
playout	4071	259	1364	1.27%	16.78%	1.57	4262	217	1478	0.71%	31.43%	1.95
ac3	631	148	300	1.53%	7.50%	0.17	718	108	366	1.39%	9.44%	0.21
xc5	2621	272	1344	3.55%	7.61%	1.02	2941	212	1665	2.69%	11.26%	1.28
hc7	2707	481	974	3.07%	3.77%	1.66	2847	303	1079	2.00%	6.10%	2.18
a9c3	4071	542	1235	1.76%	13.47%	3.56	4316	365	1450	1.05%	23.90%	4.36
pc2	12359	528	2902	2.89%	15.86%	9.51	13131	422	3455	1.84%	26.49%	12.71

example of circuit *pc2*, the area increase of BBP/RES is 2.89%, whereas that of RDM/RES is about 8.52% (2.9× larger); the *#BB* of BBP/RES is 528, whereas that of RDM/RES is about 1243 (2.35× larger). The same conclusion about the comparison of BBP versus RDM holds for the FR scenario. It is also interesting to observe that BBP algorithm does not indeed increase CPU time from RDM. Actually, it may use slightly less run time. This is because during BBP, one buffer block (not just one buffer) can be determined at a time.

- Since the FR computation/update can be computed in constant time, the run times under FR scenario only increase slightly compared to those under RES. As a result, our largest example (*pc2* with more than 13 000 buffers) only takes about 12.7 s.

To summarize, it is obvious that the BBP/FR is the best combination among these four to meet delay targets, with very marginal area increase (less than 2% for most test cases), least number of buffer blocks and comparable CPU times. It shall be noticed, however, that even under this best algorithm, there may still exist quite some nets that cannot meet their delay constraints under some given floorplan and timing budget. For this reason, and also to achieve better area ratio, it is important to have an interconnect-driven floorplanning engine to work closely with our BBP/FR algorithm. We are currently working on it.

VI. CONCLUSION AND DISCUSSIONS

In this paper, we introduced the concept of feasible region for buffer insertion and derive the analytical formula to compute FR under any given delay constraint. We then proposed an effective BBP algorithm to automatically generate buffer blocks for interconnect optimization with chip area and buffer block number minimization. Experimental results showed that our BBP/FR algorithm leads to significant improvement over previous buffer insertion/planning algorithms.

The BBP bridges the gap between interconnect layout optimization and physical design. By constructing appropriate buffer blocks and determining the rough location of each buffer for every net, we can obtain more accurate on-line interconnect estimation/prediction for wire length, congestion as well as delay with appropriate buffer insertion considered.

After our work on this topic was published in [18], there have been several followup studies. The work by [25] used a network flow formulation to compute the maximum number of buffers that can be inserted into the free space intersected with feasible regions, assuming at most one buffer for each net. The work by [26] generalized our concept of feasible region to obtain a set of independent feasible regions (IFRs), and considered both delay and congestion. But it should be mentioned that for 2-D nets, IFRs are not completely independent because of the assignments of buffers to locations within their respective 2-D IFRs must ensure a monotonic path from source to sink [27].

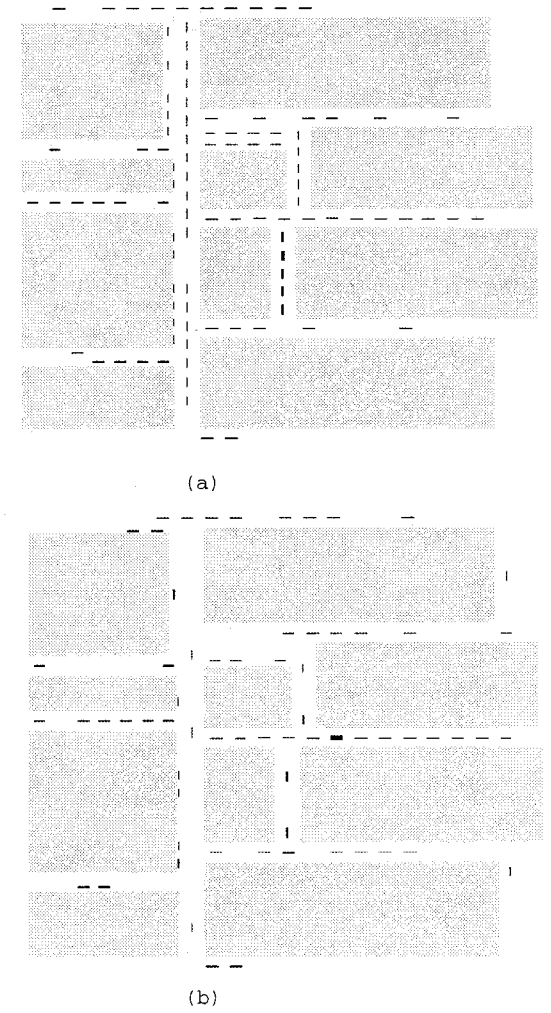

Fig. 5. Floorplan and buffer block layouts of the MCNC circuit *xerox* by (a) RDM/RES and (b) BBP/FR. The ten big blocks are circuit functional modules and the rest are buffer blocks.

The recent work by [28] studied the BBP under simultaneous delay and transition time constraints. Given an existing buffer block plan, [29] and [30] addressed the problem of how to allocate buffers to pre-existing buffer blocks. Most recently, [31] proposed to distribute buffer sites throughout the layout. Since buffer insertion is a key technique to reduce interconnect delay and noise and buffers are used extensively in high-performance designs, we expect to see more studies on efficient and effective buffer planning in the future.

ACKNOWLEDGMENT

The authors would like to thank Dr. M. K. Mohan from Intel Corporation, OR, for introducing the buffer block planning problem to them. They would also like to thank Dr. W. Donath from the IBM T. J. Watson Research Center, Yorktown Heights, NY, Dr. N. Chang from Apache Design Solutions, CA, and Dr. L. van Ginneken from Magma Design Automation, CA, for their helpful discussions.

REFERENCES

[1] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration VLSI J.*, vol. 21, pp. 1–94, 1996.
[2] J. Cong, L. He, K.-Y. Khoo, C.-K. Koh, and D. Z. Pan, "Interconnect design for deep submicron ICs," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1997, pp. 478–485.
[3] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.
[4] R. Otten, "Global wires harmful?," in *Proc. Int. Symp. Physical Design*, Monterey, CA, Apr. 1998, pp. 104–109.
[5] J. Cong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc. Asia and South Pacific Design Automation Conf.*, Hong Kong, Jan. 1999, pp. 97–100.
[6] R. Otten and R. K. Brayton, "Planning for performance," in *Proc. Design Automation Conf.*, San Francisco, CA, June 1998, pp. 122–127.
[7] J. Cong. (1997, Dec.) Challenges and opportunities for design innovations in nanometer technologies. *SRC Working Papers* [Online]. Available: http://www.src.org/prg_mgmt/frontier.dgw
[8] ——, "An interconnect-centric design flow for nanometer technologies," *Proc. IEEE*, vol. 89, pp. 505–528, Apr. 2001.
[9] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay," in *Proc. IEEE Int. Symp. Circuits and Systems*, New Orleans, LA, May 1990, pp. 865–868.
[10] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1995, pp. 138–143.
[11] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1996, pp. 44–49.
[12] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," in *Proc. Int. Symp. Physical Design*, Napa, CA, 1997, pp. 192–197.
[13] W. M. Dai, B. Eschermann, E. S. Kuh, and M. Pedram, "Hierarchical placement and floorplanning for BEAR," *IEEE Trans. Comput.-Aided Design*, vol. 8, pp. 1335–1349, Dec. 1989.
[14] D. Wong and C. L. Liu, "Floorplan design of VLSI circuits," *Algorithmica*, pp. 263–291, 1989.
[15] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1995, pp. 472–479.
[16] M. Kang, W. Dai, T. Dillinger, and D. LaPotin, "Delay bounded buffered tree construction for timing driven floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1997, pp. 707–712.
[17] J. Cong, "An interconnect-centric design flow for nanometer technologies," in *Proc. Int. Symp. VLSI Technology, Systems, and Applications*, Taiwan, June 1999, pp. 54–57.
[18] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1999, pp. 358–363.
[19] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, U.S.: Semiconductor Industry Assoc., 1997.
[20] K. Nabors and J. White, "Fastcap: A multipole accelerated 3-D capacitance extraction program," *IEEE Trans. Comput.-Aided Design*, pp. 1447–1459, Nov. 1991.
[21] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. Design Automation Conf.*, Anaheim, CA, June 1997.
[22] R. Otten, "Graphs in floor-plan design," *Int. J. Circuit Theory Applicat.*, vol. 16, pp. 391–410, Oct. 1988.
[23] [Online]. Available: http://www.cbl.ncsu.edu/cbl_docs/lys92.html
[24] J. Cong, T. Kong, D. Xu, F. Liang, J. S. Liu, and W. H. Wong, "Relaxed simulated tempering for VLSI floorplan design," in *Proc. Asia and South Pacific Design Automation Conf.*, Hong Kong, Jan. 1999, pp. 13–16.
[25] X. Tang and D.-F. Wong, "Planning buffer locations by network flows," in *Proc. Int. Symp. Physical Design*, San Diego, CA, Apr. 2000, pp. 180–185.
[26] P. Sarkar, V. Sundararaman, and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proc. Int. Symp. Physical Design*, San Diego, CA, Apr. 2000.
[27] P. Sarkar and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," *IEEE Trans. Comput.-Aided Design Integrated Circuits Syst.*, vol. 20, pp. 660–671, May 2001.
[28] ——, "Repeater block planning under simultaneous delay and transition time constraints," in *Proc. Design, Automation, and Test Eur.*, Paris, France, Mar. 2001, pp. 540–544.

[29] F. Dragan, A. Kahng, I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering using an available buffer block plan," in *Proc. Int. Conf. Computer-Aided Design*, San Diego, CA, Nov. 2000, pp. 104–109.

[30] ——, "Provably good global buffering by multiterminal multicommodity flow approximation," in *Proc. Asia and South Pacific Design Automation Conf.*, Yokohama, Japan, Jan. 2001, pp. 120–125.

[31] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. G. Villarrubia, "A practical methodology for early buffer and wire resource allocation," in *Proc. Design Automation Conf.*, Las Vegas, NV, June 2001.

Jason Cong (S'88–M'90–SM'96–F'01) received the B.S. degree in computer science from Peking University, Beijing, China, in 1985 and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, in 1987 and 1990, respectively.

Currently, he is a Professor and Co-Director of the VLSI CAD Laboratory, Computer Science Department, University of California, Los Angeles. He was appointed as a Guest Professor of Peking University in 2000. His research interests include layout synthesis and logic synthesis for high-performance low-power VLSI circuits, design and optimization of high-speed VLSI interconnects, field programmable gate arrays (FPGAs) synthesis and reconfigurable architectures and has published over 150 research papers and led over 20 research projects supported by DARPA, NSF, and a number of industrial sponsors in these areas. He is an Associate Editor of *ACM Transactions on Design Automation of Electronic Systems*.

Dr. Cong has served as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposium on FPGAs, respectively, and the Program Co-Chair of the 1999 International Symposium on Low-Power Electronics and Designs. He has also served on program committees of many major conferences including DAC, ICCAD, and ISCAS. He is an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. He received the Best Graduate Award from Peking University in 1985 and the Ross J. Martin Award for Excellence in Research from the University of Illinois at Urbana-Champaign in 1989. He received the NSF Young Investigator Award in 1993, the Northrop Outstanding Junior Faculty Research Award from UCLA in 1993, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award in 1995 from the IEEE CAS Society, the ACM SIGDA Meritorious Service Award in 1998, an SRC Inventor Recognition Award in 2000, and the SRC Technical Excellence Award in 2001.

Tianming Kong (S'99) received the M.S. and B.S. degrees in computer science from Tsinghua University, Beijing, China, in 1993 and 1997, respectively. He is currently working toward the Ph.D. degree in computer science at the University of California, Los Angeles.

His research interest is focused on very large scale integrated (VLSI) physical design and global local optimization methods.

Zhigang (David) Pan (S'97–M'00) received the B.S. degree in geophysics from Peking University in 1992, the M.S. degree in atmospheric sciences, the M.S. degree in computer science, and the Ph.D. degree in computer science, all from University of California at Los Angeles (UCLA) in 1994, 1998, and 2000, respectively.

He was with Magma Design Automation, Inc., Cupertino, CA, during the summer of 1999, and with the IBM T. J. Watson Research Center, Yorktown Heights, NY, during the summer of 2000. He is currently a Research Staff Member at the IBM T. J. Watson Research Center. His current research interests include very large scale integration (VLSI) interconnect modeling, synthesis, planning and their interaction with physical design, logic synthesis, and register-transfer level (RTL) planning. He is an Industrial Mentor to several SRC-sponsored university research projects.

Dr. Pan is the Local Arrangement Chair and serves in the Technical Program Committee of the 12th ACM Great Lakes Symposium on VLSI. He received the Best Paper in Session Award from SRC Techcon 1998, the IBM Research Fellowship in 1999, the Dimitris Chorafas Foundation Prize (from Switzerland) in 2000, the SRC Inventor Recognition Award in 2000, and the UCLA Outstanding Ph.D. Award in 2001.