

Simultaneous Floorplan and Buffer-Block Optimization

Iris Hui-Ru Jiang, Yao-Wen Chang, *Member, IEEE*, Jing-Yang Jou, *Senior Member, IEEE*, and Kai-Yuan Chao

Abstract—As technology advances and the number of interconnections among modules rapidly increases, timing closure, and design convergence are the most important concerns. Hence, it is desirable to consider interconnect optimization as early as possible. Previous work for this issue can be classified into two directions: wire planning and buffer-block planning for interconnect-driven floorplanning. Wire planning for interconnect-driven floorplanning does not consider buffer insertion, and buffer-block planning for interconnect-driven floorplanning cannot overcome the limitation of a bad initial floorplan. In this paper, we first address simultaneous floorplanning and buffer-block planning (i.e., integrating buffer-block planning into floorplanning) for interconnect optimization. We adopt simulated annealing to refine a floorplan so that buffers can be inserted more effectively. In each iteration, we construct a routing tree for each net, allocate buffers for all nets, introduce corresponding buffer blocks into the intermediate floorplan, and invoke Lagrangian relaxation to optimize area and satisfy timing requirements. Further, in order to reduce the problem size, we present supermodule partitioning which partitions modules into supermodules. Experimental results show that our method of integrating buffer-block planning into floorplanning can significantly improve the interconnect delay and reduce the number of buffers needed. Based on a set of MCNC benchmark circuits, our approach achieves an average success rate of 86.1% of nets meeting timing constraints, inserts only 272 buffers on average, and consumes an average extra area of only 0.28% over the given floorplan, compared with the average success rate of 62.6%, 1123 buffers, and extra area of 1.05% resulted from a famous recent work presented at ICCAD'99.

Index Terms—Floorplanning, interconnect optimization, layout, physical design.

I. INTRODUCTION

AS REVEALED by the 1999 *International Technology Roadmap for Semiconductors* [13], technology will soon shrink into below 0.1 μm and the chip complexity will be over 200 million transistors soon. For such large and complex designs, timing closure and design convergence are the most important concerns. Further, for deep submicron designs, interconnect dominates circuit performance. However, the

conventional design flow deals with interconnect optimization at the routing or the postrouting stage. When the amount of communication among modules rapidly increases, it is almost impossible to remedy interconnect during or after routing, since most silicon and routing resources are occupied. Therefore, we should optimize interconnect as early as possible. Previous work for this issue can be classified into two directions: wire planning and buffer-block planning for interconnect-driven floorplanning.

Wire planning for interconnect-driven floorplanning tries to measure the impact of wiring or to plan interconnect at the floorplanning stage [4]. However, this method considers only wires; other useful techniques, e.g., buffer insertion, were not included. On the other hand, buffer-block planning for interconnect-driven floorplanning manages buffer-blocks for a given floorplan [5], [11], [14]. Previous work has shown that buffer insertion is an effective and widely used technique to improve interconnect delay, especially for global signals [1], [13]. (For example, over 85% of global nets in Intel Itanium microprocessors are buffered to reshape signals [9].) Because buffers consume silicon resource, it is too difficult to insert a large number of buffers individually after placement or routing when most silicon and routing resources are occupied. The induced area may significantly change the floorplan and placement, thus causing problems in timing closure and design convergence. To tackle this problem, researchers tried to consider buffer insertion during postfloorplanning (not during routing or postrouting) [5], [11], [14]. For a given floorplan, channels and dead spaces are used as buffer blocks, which accommodate buffers. Cong *et al.* first consider this issue in [5]; they derive feasible region formulas to determine where to insert buffers to meet timing requirements and propose a greedy algorithm to plan buffer blocks in a slicing floorplan. Sarkar *et al.* also consider routability and address the concept of independent feasible regions (feasible regions of buffers for a net do not influence each other) in [11]. Tang and Wong optimally plan as many buffers into buffer blocks as possible for all nets, each with one buffer in [14]. Moreover, [5] and [11] expand channels to provide more buffers, if necessary. However, if the given floorplan is not good enough, channel expansion would result in much area overhead. Hence, this kind of strategy is limited by the quality of a given floorplan. Although [5] claims their approach can be applied to slicing and nonslicing floorplans, channel expansion can be adopted only when the channel definition is certain. For slicing floorplans, each channel is explicitly shown in the representation, e.g., slicing floorplan trees [16]. However, a channel may implicitly be defined in a nonslicing floorplan. Hence, channel

Manuscript received July 8, 2002; revised January 29, 2003. This work was supported in part by the National Science Council of Taiwan, R.O.C., under Grant NSC 91-2215-E-002-038. This paper was recommended by Associate Editor M. D. F. Wong.

I. H.-R. Jiang is with VIA Technologies Inc., Taipei 231, Taiwan (e-mail: huiru@cis.nctu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

J.-Y. Jou is with the Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: jyjou@ee.nctu.edu.tw).

K.-Y. Chao is with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: kchao@ichips.intel.com).

Digital Object Identifier 10.1109/TCAD.2004.826582

expansion cannot easily be applied to nonslicing floorplans. Alpert *et al.* proposed buffer-site methodology in [3], allocating buffers into empty silicon area inside macroblocks. However, placing buffers inside macroblocks requires one to consider the interaction between logic and interconnect. Therefore, buffers are typically inserted outside macroblocks [9].

Previous work for interconnect-driven floorplanning does not integrate buffer insertion into floorplanning. Existing work for buffer-block planning for interconnect-driven floorplanning cannot break through the limitation by a bad floorplan. In this paper, we first study simultaneous floorplanning and buffer-block planning (FBP) to conquer the weakness of the above. (In industry, this idea was considered for Intel Itanium microprocessor design [9].) We present an algorithm that simultaneously considers FBP for a general floorplan. Our method adopts the simulated annealing mechanism to refine the floorplan so that buffers can be inserted more effectively. In each iteration, we construct a routing tree for each net, allocate buffers for all nets, introduce corresponding buffer blocks into the intermediate floorplan, and invoke Lagrangian relaxation to optimize area and satisfy timing requirements. Further, in order to reduce the problem size, we present supermodule partitioning which partitions modules into supermodules.

Experimental results show that our method of integrating buffer-block planning into floorplanning can significantly improve the interconnect delay and reduce the number of buffers needed. Based on a set of MCNC benchmark circuits, our approach achieves an average success rate of 86.1% of nets meeting timing constraints, insert only 272 buffers on average, and consumes an average extra area of only 0.28% over the given floorplan, compared with the average success rate of 62.6%, 1123 buffers, and extra area of 1.05% resulted from the recent work in [5].

The rest of this paper is organized as follows. Section II gives the problem formulation of simultaneous FBP. Section III introduces the concept of a nonslicing floorplan representation, independent feasible regions, and basic buffer-block planning. We detail Lagrangian relaxation-based buffer-block planning and supermodule partitioning in Section IV and the simulated annealing algorithm in Section V. Experimental results are discussed in Section VI. Section VII concludes this paper.

II. PROBLEM FORMULATION

In this section, we give our problem formulation. We define the simultaneous FBP problem as follows.

- **Problem:** The simultaneous FBP problem.
- **Objective:** Minimize area overhead, subject to timing requirements.
- **Inputs:** An initial floorplan, multiterminal nets, and their timing requirements, buffer library, technology file.
- **Outputs:** A floorplan with buffer-block planning.

Table I lists the technology file and buffer library used in our experiments that are based on 0.18- μm technology in the NTRS'97 roadmap [12]. These parameters were also used in [5] and [11]. The notation is used throughout this paper.

TABLE I
PARAMETERS OF 0.18- μm TECHNOLOGY IN THE NTRS'97 ROADMAP

Parameter	Description (unit)	Value
r_{\square}	wire sheet resistance (Ω/\square)	0.068
\hat{r}	wire unit-length resistance of 0.9 μm width ($\Omega/\mu\text{m}$)	0.075
c_a	wire sheet area capacitance ($fF/\mu\text{m}^2$)	0.06
c_f	wire fringing capacitance ($fF/\mu\text{m}$)	0.064
ω	wire width (μm)	0.9
\hat{c}	wire unit-length capacitance of 0.9 μm width ($fF/\mu\text{m}$)	0.118
C^L	load capacitance (fF)	23.4
R^D	driver resistance (Ω)	180
D_b	intrinsic buffer delay (ps)	36.4
C_b	buffer input capacitance (fF)	23.4
R_b	buffer output resistance (Ω)	180
A_b	buffer size (μm^2)	400

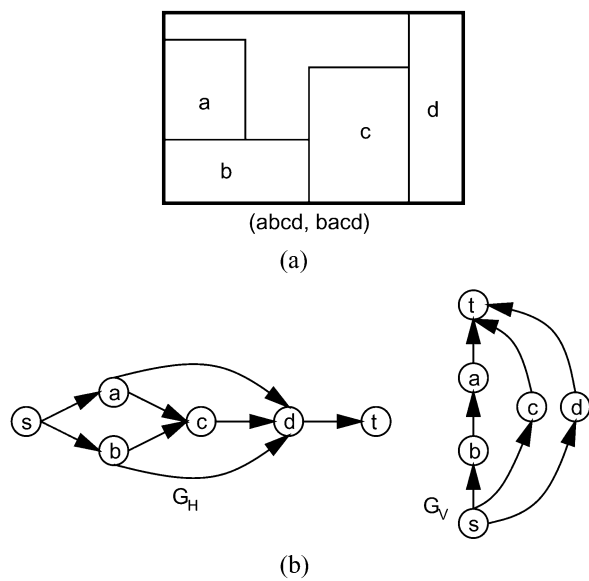


Fig. 1. (a) Packing of a sequence pair $(abcd, bacd)$ for modules $\{a, b, c, d\}$. (b) The corresponding horizontal and vertical constraint graphs.

III. PRELIMINARIES

This section first introduces the sequence-pair representation of a nonslicing floorplan [10] and the concepts of independent feasible regions [11]. We then propose our approach for buffer-block planning on two-terminal nets.

A. Sequence-Pair Representation

We adopt the sequence-pair representation [10] for a general floorplan. A sequence pair of a set of modules is a pair of sequences formed by module names. For example, given a set of modules $\{a, b, c, d\}$, $(abcd, bacd)$ is a sequence pair of these modules, as shown in Fig. 1(a). Based on the following properties, we can retrieve the topology relations between modules.

- **H-constraint:** If $(\dots a \dots b \dots, \dots a \dots b \dots)$, module b is on the right side of module a .
- **V-constraint:** If $(\dots a \dots b \dots, \dots b \dots a \dots)$, module b is below module a .

We can accordingly construct the horizontal and vertical constraint graphs, G_H and G_V . In G_H/G_V , we construct a node for each module and two additional nodes s and t . Except s and t

whose weights are zero, each node in G_H/G_V is weighted as the width/height of the corresponding module. The edges are constructed by the following rules.

- There exists an edge (a, b) from a to b in G_H iff $(\dots a \dots b \dots, \dots a \dots b \dots)$.
- There exists an edge (a, b) from b to a in G_V iff $(\dots a \dots b \dots, \dots b \dots a \dots)$.

In addition, edges from s to zero-indegree nodes and from zero-outdegree nodes to t are added. Fig. 1(b) illustrates the corresponding constraint graphs of the sequence pair $(abcd, bacd)$. The x -coordinate (y -coordinate) of the bottom-left corner of each module can be computed by the longest path length from s to the module node in G_H (G_V). Hence, if a dummy module replaces t and an additional edge from the dummy module to t is added in G_H (G_V), the x -coordinate (y -coordinate) of the bottom-left corner of the dummy module equals the width (height) of the packing. By a sequence of the following two kinds of perturbations, an arbitrary sequence pair can change to a given one.

- Exchange two modules in the first sequence.
- Exchange two modules in both sequences.

B. Independent Feasible Region

In this section, we present the computation of independent feasible regions proposed by [11]. The independent feasible region of a buffer is the region where the buffer can be placed to meet the timing requirement of the net, while the other buffers are placed within their respective independent feasible regions.

Given a wire segment of length l with driver resistance R^D , load capacitance C^L , wire resistance per unit length \hat{r} , and wire capacitance per unit length \hat{c} , its Elmore delay is calculated by

$$D(R^D, C^L, l) = \frac{\hat{r}\hat{c}l^2}{2} + (R^D\hat{c} + \hat{r}C^L)l + R^D C^L.$$

Assume that R_b is the buffer output resistance, and C_b is the buffer input capacitance. Let $D_{\text{net}}^j(l_1, l_2, \dots, l_n)$ denote the Elmore delay of a two-terminal net j of length l with n buffers inserted, where l_i is the distance between the driver and the i th buffer. The buffer locations under the optimal delay $D_{\text{opt}}^j = D_{\text{net}}^j(l_1^*, l_2^*, \dots, l_n^*)$ are

$$l_i^* = \kappa_1 + (i-1)\kappa_2, \quad i \in \{1, 2, \dots, n\}$$

where

$$\kappa_1 = \frac{1}{n+1} \left(l + \frac{n(R_b - R^D)}{\hat{r}} + \frac{(C^L - C_b)}{\hat{c}} \right)$$

$$\kappa_2 = \frac{1}{n+1} \left(l - \frac{(R_b - R^D)}{\hat{r}} + \frac{(C^L - C_b)}{\hat{c}} \right).$$

The width of the independent feasible region of a buffer means the maximum tolerable range around the optimum location of the buffer. In [11], the independent feasible region F_i^j of width W_F^j for the i th buffer of a net j is defined as

$$F_i^j = \left(l_i^* - \frac{W_F^j}{2}, l_i^* + \frac{W_F^j}{2} \right) \cap (0, l)$$

such that $(l_1, l_2, \dots, l_n) \in F_1^j \times F_2^j \times \dots \times F_n^j$ and $D_{\text{net}}^j(l_1, l_2, \dots, l_n) \leq D_{\text{req}}^j$, where D_{req}^j denotes the timing requirement associated with net j . Moreover, if $D_{\text{req}}^j \geq D_{\text{opt}}^j$,

the width W_F^j of the independent feasible region for each buffer of net j is

$$W_F^j = 2\sqrt{\frac{D_{\text{req}}^j - D_{\text{opt}}^j}{\hat{r}\hat{c}(2n-1)}}.$$

On the other hand, in [5], the minimum number n_{min}^j of buffers required to meet the timing requirement D_{req}^j for a net j of length l is

$$n_{\text{min}}^j = \left\lceil \frac{O_2 - \sqrt{O_2^2 - 4O_1O_3}}{2O_1} \right\rceil$$

where

$$O_1 = R_b C_b + D_b$$

$$O_2 = D_{\text{req}}^j + \frac{\hat{r}}{\hat{c}}(C_b - C^L)^2 + \frac{\hat{c}}{\hat{r}}(R_b - R^D)^2 - (\hat{r}C_b + \hat{c}R_b)l - D_b - R^D C_b - R_b C^L$$

$$O_3 = \frac{\hat{r}\hat{c}l^2}{2} + (\hat{r}C^L + \hat{c}R^D)l - D_{\text{req}}^j.$$

C. Basic Buffer-Block Planning

In this section, we propose the basic idea of our buffer-block planning for two-terminal nets. (Multiterminal nets will be considered later.) Fig. 2(a) shows the independent feasible regions of two buffers on a two-terminal net j . Based on the formulas shown in the previous subsection, the routing of a two-terminal net should be a monotonic route restricted in the bounding box of its terminals. The independent feasible region of the i th buffer is a hexagon or a degenerated hexagon bounded by the bounding box and two parallel lines of slope $+1$ or -1 . The respective distance from the source terminal to these parallel lines are $l_i^* - W_F^j/2$ and $l_i^* + W_F^j/2$.

A buffer block is a rectangular region consisting of buffers, provided by dead spaces and/or channels. As shown in Fig. 2(a), each buffer is inserted into a buffer block with which its independent feasible region overlaps. For the first buffer, its independent feasible region intersects the dead space f , thus, it is assigned to the buffer block f . If there are many choices, we first assign it to the one with the most overlapped area. For the second buffer, there is no dead space intersecting its independent feasible region, thus it is assigned to the channel h (between modules c and d), which is nearest to its independent feasible region. After all buffers for all nets are allocated, the region of each buffer block is determined as the bounding rectangle of the inserted buffers. We then treat a buffer block as a soft module, and insert the node into the constraint graphs accordingly. See Fig. 2(b) for an illustration. Since we remove all transitive edges before processing, inserting a buffer-block node into the constraint graph needs only linear time. We will reshape the floorplan by Lagrangian relaxation detailed in Section IV.

IV. LAGRANGIAN RELAXATION-BASED BUFFER-BLOCK PLANNING

In this section, we detail buffer-block planning for an intermediate floorplan. We construct a routing tree for each net, assign buffer blocks (extended from the basic idea introduced in

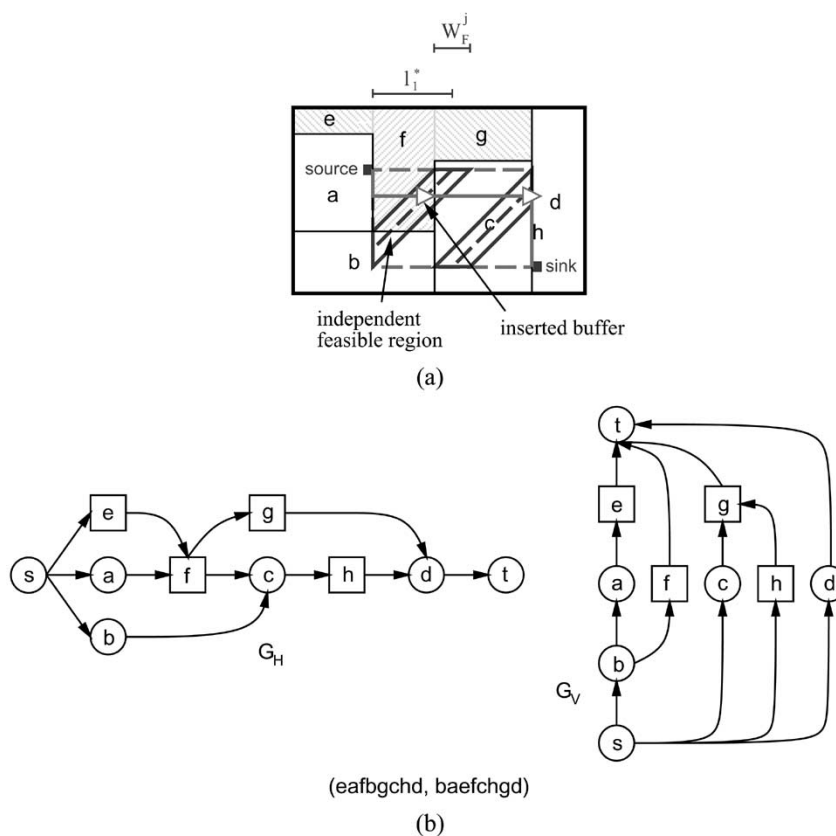


Fig. 2. (a) Net requires two buffers. Each buffer can be inserted into its independent feasible region. In the case shown in this figure, one buffer is inserted to the dead space f , the other is inserted to the channel h (on the right side of module c). (b) The modified sequence pair with induced buffer blocks and its corresponding constraint graphs, where transitive edges are not shown, and induced buffer-block nodes are indicated by rectangles.

Section III), reshape the floorplan using the Lagrangian relaxation technique, partition the floorplan into supermodules, and, finally, summarize our buffer-block planning procedure.

A. Routing Tree Construction

For an intermediate floorplan, we first construct a routing tree for each multiterminal net. At the floorplanning stage, detailed timing information is not available. Thus, our goal is to construct a timing-aware routing tree for each net.

We adopt the AHHK heuristic presented by [2] to combine Dijkstra's shortest path algorithm with Prim's minimum spanning tree one [6]. The generated tree directly tradeoffs between radius and wire length. The initial tree is then converted to a Steiner tree by removing overlapped edges based on the algorithm proposed in [7]. Fig. 3(a) shows an example of a multiterminal routing tree, the longest path (source \rightarrow sink2 \rightarrow sink3) is indicated by the bold line. (Alternative tree construction approaches can also be used instead.) Based on the formulas described in Section III-B, we can check whether an optimal buffered routing tree can satisfy its timing requirement, i.e., $D_{\text{opt}}^i \leq D_{\text{req}}^i$. We record these unsatisfied nets, which do not meet timing requirements, even with optimally inserted buffers, and do not plan buffers for them (since the timing of those nets cannot be satisfied).

B. Buffer-Block Planning

A multiterminal routing tree can be seen as a combination of several two-terminal routing segments. Hence, our buffer-block planning for multiterminal nets is extended from the basic buffer-block planning for two-terminal nets presented in Section III-C.

After checking whether a routing tree can satisfy its timing requirement, we record unsatisfied nets and do not plan buffers for them. For the rest of the nets, we process path by path (from the longest to the shortest) in each routing tree. Based on the formulas in Section III-B, we obtain the number of buffers needed for the longest path, the optimal distance from the source terminal to each buffer, and the width of independent feasible region. We then determine the independent feasible region of each buffer on each path according to the above information.

Fig. 3(b) shows the independent feasible regions of buffer assignment for the routing tree given in Fig. 3(a). In this case, the longest path (source \rightarrow sink2 \rightarrow sink3) requires two buffers, and the path from the source to sink1 does not need buffers. To preserve the topology, the independent feasible region of each buffer is further restricted to the bounding box of the two nearest Steiner tree nodes. If the independent feasible region covers some tree node, the tree node plays the role of the buffer. As shown in Fig. 3(b), the independent feasible region of the first buffer is subject to the nearest tree nodes, and the second

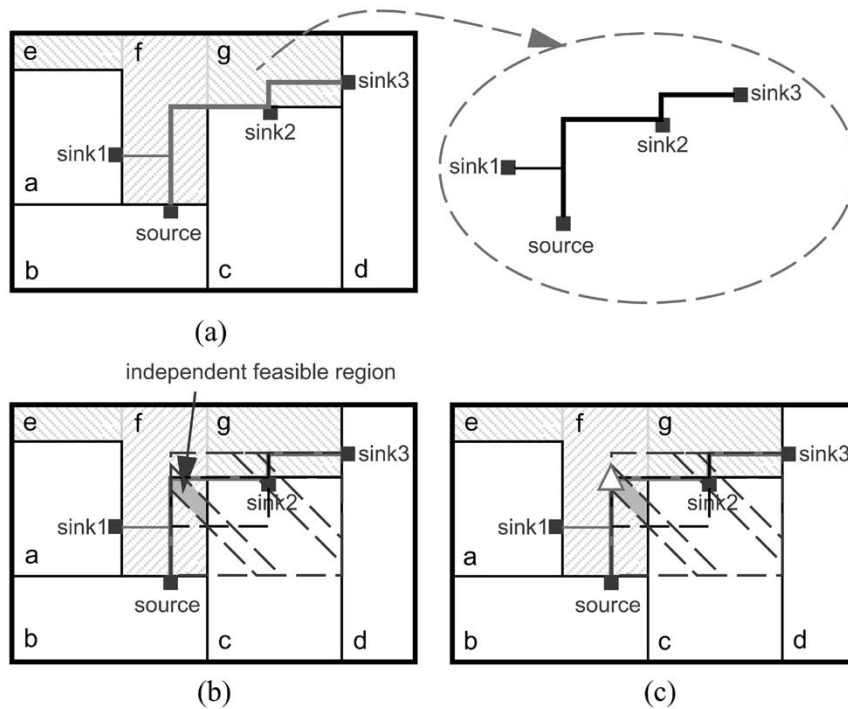


Fig. 3. (a) The routing tree for a multiterminal net, where the longest path (source \rightarrow sink2 \rightarrow sink3) is highlighted by the bold line. We process the tree path by path, from the longest to the shortest. (b) The path from the source to sink3 requires two buffers; the corresponding independent feasible region of the first buffer is shown by the shaded hexagon, and the second buffer is covered by sink2. (c) The resulting buffer assignment for the longest path; the first buffer is assigned to the buffer block f , and the second buffer is taken by sink2.

buffer is replaced by the sink2 terminal. Similar to the basic buffer-block planning for two-terminal nets, we assign buffers into a dead space that intersects their independent feasible regions with the most area or into the nearest channel; as shown in Fig. 3(c), the first buffer is assigned to the buffer block f .

After allocating buffers for all nets, we introduce buffer blocks as soft modules into constraint graphs. These buffer blocks may occupy dead spaces or be inserted into channels. Their areas equal the bounding areas of inserted buffers. Previous work generates buffer blocks *before* buffer assignment; however, we generate buffer blocks *after* buffer assignment and, thus, the area of buffer blocks can properly be controlled, especially for the buffer blocks in channels.

C. Lagrangian Relaxation

We adopt the Lagrangian relaxation technique to reshape the floorplan. After buffer allocation, G_H/G_V contains m modules nodes and b buffer-block nodes. The first m nodes indicate modules, and the other b nodes indicate buffer blocks. Each module or buffer block has its bottom-left corner x -coordinate x_i , bottom-left corner y -coordinate y_i , area A_i , width w_i , height A_i/w_i , maximum width U_i , and minimum width L_i . In addition, inspired by [17] to facilitate area calculation, we add one dummy node labeled $m + b + 1$ to G_H and G_V . As indicated in Fig. 4(b), each edge directed to t is altered to the dummy node, and an additional edge from $m + b + 1$ to t is added. As mentioned in Section III-A, x_{m+b+1} (y_{m+b+1}) equals the width (height) of the packing. There are n multiterminal nets. D_{req}^i denotes the timing requirement of net i , and D_{net}^i denotes the longest path delay in the routing tree of net i .

Hence, we may formulate the geometric program \mathcal{PP} (primal problem) to minimize the total area subject to timing requirements as follows.

$$\begin{aligned}
 &\text{Minimize} && x_{m+b+1}y_{m+b+1} \\
 &\text{Subject to} && x_i + w_i \leq x_j, && \forall (i,j) \in G_H \\
 &&& y_i + \frac{A_i}{w_i} \leq y_j, && \forall (i,j) \in G_V \\
 &&& D_{\text{net}}^i \leq D_{\text{req}}^i, && \forall 1 \leq i \leq n \\
 &&& L_i \leq w_i \leq U_i, && \forall 1 \leq i \leq m + b.
 \end{aligned}$$

Because the objective function and the constraints are all posynomial [15], we can apply Lagrangian relaxation to solve the problem \mathcal{PP} by introducing one nonnegative Lagrange multiplier for each constraint. Therefore, the Lagrangian relaxation subproblem \mathcal{LRS} is given by

$$\begin{aligned}
 &\text{Minimize} && x_{m+b+1}y_{m+b+1} \\
 &&& + \sum_{(i,j) \in G_H} \lambda_{i,j}(x_i + w_i - x_j) \\
 &&& + \sum_{(i,j) \in G_V} \mu_{i,j} \left(y_i + \frac{A_i}{w_i} - y_j \right) \\
 &&& + \sum_{1 \leq i \leq n} \gamma_i (D_{\text{net}}^i - D_{\text{req}}^i) \\
 &\text{Subject to} && L_i \leq w_i \leq U_i, \quad \forall 1 \leq i \leq m + b.
 \end{aligned}$$

The objective function of \mathcal{LRS} is the Lagrangian function $L_{\lambda,\mu,\gamma}$. We have the following theorem to simplify the Lagrangian function.

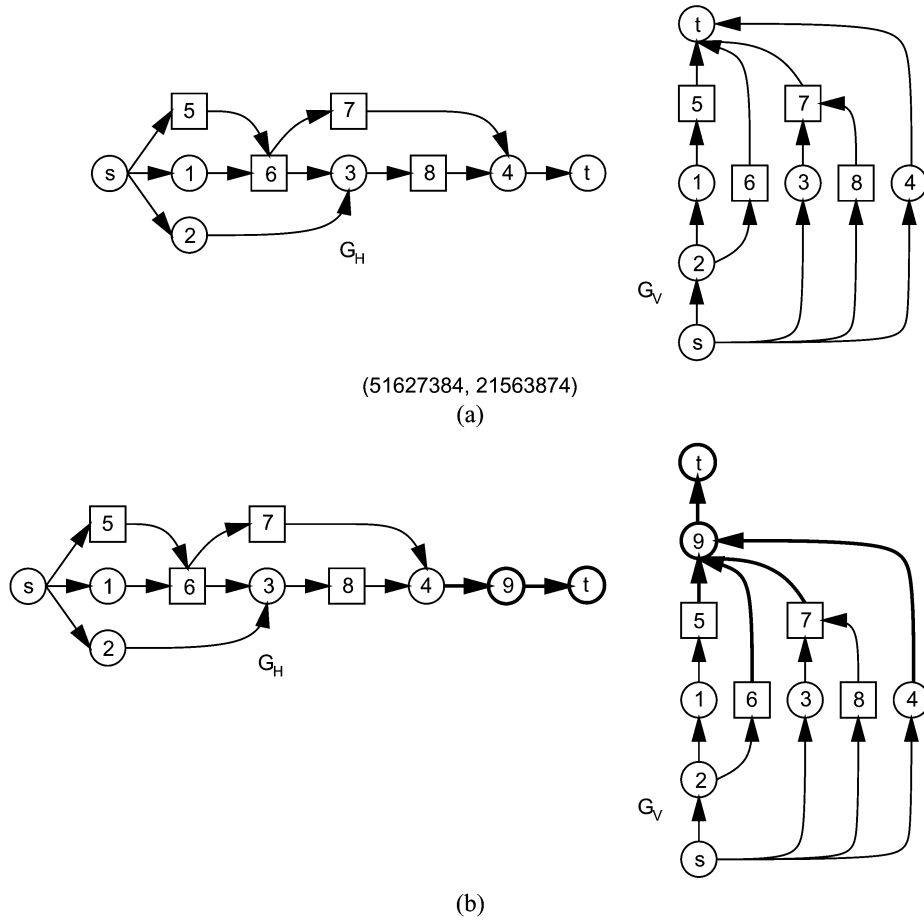


Fig. 4. (a) Original constraint graphs, G_H/G_V , where nodes 1–4 are modules, and nodes 5–8 are induced buffer blocks. (b) The modified constraint graphs added with the dummy node 9, where the modification is highlighted by bold lines.

Theorem 1: The optimality conditions for the Lagrange multipliers are given by

$$\begin{aligned} \sum_{(j,i) \in G_H} \lambda_{j,i} &= \sum_{(i,j) \in G_H} \lambda_{i,j}, \quad \forall 1 \leq i \leq m+b \\ \sum_{(j,i) \in G_V} \mu_{j,i} &= \sum_{(i,j) \in G_V} \mu_{i,j}, \quad \forall 1 \leq i \leq m+b \\ x_{m+b+1} &= \sum_{(i,m+b+1) \in G_V} \mu_{i,m+b+1} \\ y_{m+b+1} &= \sum_{(i,m+b+1) \in G_H} \lambda_{i,m+b+1}. \end{aligned}$$

Proof: By Kuhn–Tucker conditions [15], the first order derivative of $L_{\lambda,\mu,\gamma}$, with respect to each variable, equals 0 at the optimal solution of \mathcal{PP} .

Rearranging $L_{\lambda,\mu,\gamma}$, we have

$$\begin{aligned} L_{\lambda,\mu,\gamma} &= x_{m+b+1}y_{m+b+1} \\ &\quad - \sum_{(i,m+b+1) \in G_H} \lambda_{i,m+b+1}x_{m+b+1} \\ &\quad - \sum_{(i,m+b+1) \in G_V} \mu_{i,m+b+1}y_{m+b+1} \\ &\quad + \sum_{1 \leq i \leq m+b} \left(\sum_{(i,j) \in G_H} \lambda_{i,j} - \sum_{(j,i) \in G_H} \lambda_{j,i} \right) x_i \end{aligned}$$

$$\begin{aligned} &+ \sum_{1 \leq i \leq m+b} \left(\sum_{(i,j) \in G_V} \mu_{i,j} - \sum_{(j,i) \in G_V} \mu_{j,i} \right) y_i \\ &+ \sum_{1 \leq i \leq m+b} \left(\sum_{(i,j) \in G_H} \lambda_{i,j}w_i + \sum_{(i,j) \in G_V} \mu_{i,j} \frac{A_i}{w_i} \right) \\ &+ \sum_{1 \leq i \leq n} \gamma_i (D_{\text{net}}^i - D_{\text{req}}^i). \end{aligned}$$

By checking Kuhn–Tucker conditions, this theorem thus follows. ■

Applying the optimality conditions, we may further simplify $L_{\lambda,\mu,\gamma}$ as follows:

$$\begin{aligned} L_{\lambda,\mu,\gamma} &= - \sum_{(i,m+b+1) \in G_H} \lambda_{i,m+b+1} \sum_{(i,m+b+1) \in G_V} \mu_{i,m+b+1} \\ &\quad - \sum_{1 \leq i \leq n} \gamma_i D_{\text{req}}^i \\ &\quad + \sum_{1 \leq i \leq m+b} \left(\sum_{(i,j) \in G_H} \lambda_{i,j}w_i + \sum_{(i,j) \in G_V} \mu_{i,j} \frac{A_i}{w_i} \right) \\ &\quad + \sum_{1 \leq i \leq n} \gamma_i D_{\text{net}}^i \end{aligned}$$

where

$(\sum_{(i,m+b+1) \in G_H} \lambda_{i,m+b+1})(\sum_{(i,m+b+1) \in G_V} \mu_{i,m+b+1})$ and $\sum_{1 \leq i \leq n} \gamma_i D_{\text{req}}^i$ are constant for a fixed vector of Lagrange multipliers.

Theorem 2: Let (w_1, \dots, w_{m+b}) be a solution, then the optimal width of module or buffer block i is given by

$$w_i^* = \min \left(U_i, \max \left(L_i, \sqrt{\frac{A_i \sum_{(i,j) \in G_V} \mu_{i,j}}{\sum_{(i,j) \in G_H} \lambda_{i,j}}} \right) \right).$$

The optimal Manhattan distance between the buffer at j and the buffer at k of net $i, l_{i,k,q}^*$, is constrained by

$$\hat{r} \hat{c} l_{i,j,k}^* + R_j \hat{c} + \hat{r} C_k = \hat{r} \hat{c} l_{i,k,q}^* + R_k \hat{c} + \hat{r} C_q$$

$\forall 1 \leq i \leq n, (j, k)$ and (k, q) are consecutive edges in the longest path ϕ_i of net i .

Proof: Differentiating $L_{\lambda, \mu, \gamma}$ with respect to w_i , we have

$$\sum_{(i,j) \in G_H} \lambda_{i,j} - \sum_{(i,j) \in G_V} \mu_{i,j} \frac{A_i}{w_i^2} = 0$$

$$w_i = \sqrt{\frac{A_i \sum_{(i,j) \in G_V} \mu_{i,j}}{\sum_{(i,j) \in G_H} \lambda_{i,j}}}.$$

Applying the range constraints on width, we have the optimal width

$$w_i^* = \min \left(U_i, \max \left(L_i, \sqrt{\frac{A_i \sum_{(i,j) \in G_V} \mu_{i,j}}{\sum_{(i,j) \in G_H} \lambda_{i,j}}} \right) \right).$$

The delay of net i, D_{net}^i , is given by

$$D_{\text{net}}^i = \sum_{(j,k) \in \phi_i} D(R_j, C_k, l_{i,j,k}) + (|\phi_i| - 1) D_b$$

where ϕ_i is the longest path in the routing tree of net i , ϕ_i has $|\phi_i|$ segments, thus, $|\phi_i| - 1$ buffers inserted, R_j is the driver resistance at j , C_k is the load capacitance at k , $l_{i,j,k}$ is the Manhattan distance between the buffer at j and the buffer at k of net i , $D(R_j, C_k, l_{i,j,k})$ is the delay associated with the edge (j, k) , and D_b is the buffer delay.

We assume that a sink terminal of a net can be a driver for other sink terminals, and the driver delay of the sink terminal equals the buffer delay D_b . Therefore, the timing constraints $D_{\text{net}}^i \leq D_{\text{req}}^i \forall 1 \leq i \leq n$ can be rewritten as

$$\sum_{(j,k) \in \phi_i} D(R_j, C_k, l_{i,j,k}) + (|\phi_i| - 1) D_b \leq D_{\text{req}}^i, \quad \forall 1 \leq i \leq n$$

where

$$D(R_j, C_k, l_{i,j,k}) = \left(\frac{\hat{r} \hat{c}}{2} \right) l_{i,j,k}^2 + (R_j \hat{c} + \hat{r} C_k) l_{i,j,k} + R_j C_k.$$

For two consecutive edges (j, k) and (k, q) in ϕ_i

$$\frac{\partial L_{\lambda, \mu, \gamma}}{\partial l_{i,j,k}} = - \frac{\partial L_{\lambda, \mu, \gamma}}{\partial l_{i,k,q}}.$$

Since the first order derivative of $L_{\lambda, \mu, \gamma}$ with respect to $l_{i,j,k}$ equals 0, we have

$$\hat{r} \hat{c} l_{i,j,k}^* + R_j \hat{c} + \hat{r} C_k = \hat{r} \hat{c} l_{i,k,q}^* + R_k \hat{c} + \hat{r} C_q.$$

$\forall 1 \leq i \leq n, (j, k)$ and (k, q) are consecutive edges in ϕ_i . This theorem thus follows. ■

The Lagrangian dual problem (\mathcal{LDP}) is to find a vector of Lagrange multipliers such that the optimal solution of \mathcal{LRS} is also the optimal solution of \mathcal{PP} .

$$\begin{aligned} & \text{Maximize} && Q(\lambda, \mu, \gamma) \\ & \text{Subject to} && \lambda, \mu, \gamma \text{ in the optimality conditions} \end{aligned}$$

where

$$Q(\lambda, \mu, \gamma) = \min L_{\lambda, \mu, \gamma}.$$

We only need to consider those multipliers satisfying the optimality conditions. We iteratively adjust multipliers by the sub-gradient optimization method as follows:

$$\begin{aligned} \lambda'_{i,j} &= [\lambda_{i,j} + \theta_k (x_i + w_i - x_j)]^+ \\ \mu'_{i,j} &= \left[\mu_{i,j} + \theta_k \left(y_i + \frac{A_i}{w_i} - y_j \right) \right]^+ \\ \gamma'_i &= [\gamma_i + \theta_k (D_{\text{net}}^i - D_{\text{req}}^i)]^+ \end{aligned}$$

where $[x]^+ = \max(0, x)$ and $\langle \theta_k \rangle$ is the step-size sequence that satisfies $\lim_{k \rightarrow \infty} \theta_k = 0$ and $\sum_{k=1}^{\infty} \theta_k = \infty$ (e.g., $\theta_k = 1/k$). After applying the subgradient optimization method, Lagrange multipliers change to a new vector, thus, the new vector needs to be projected back to the nearest point by the 2-norm measure and to meet the optimality conditions.

D. Supermodule Partitioning

After Lagrangian relaxation, we partition the floorplan into supermodules to reduce the problem size for simulated annealing. At a high temperature, the size of a supermodule is small so that the simulated annealing can freely refine the floorplan. When the temperature is cooling down (the floorplan is settled down at a low temperature), the size of a supermodule is adjusted to a larger value. A supermodule holds the following two properties.

- A supermodule is a set of modules in the floorplan.
- The nets between any pair of modules in a supermodule meet timing requirements.

An extreme case is all modules in one supermodule, i.e., all nets meet timing requirements. Note that buffer blocks in a supermodule will be considered for buffer-block planning in the next iteration, and supermodules are considered as hard modules. Fig. 5 summarizes the procedure of supermodule partitioning.

E. Summary on Buffer-Block Planning

Fig. 6 lists our buffer-block planning procedure. In lines 1 and 2, constraint graphs are extracted according to the given intermediate floorplan, and transitive edges are deleted. In lines 4–7, the routing trees are then constructed, and unsatisfied nets are recorded. In lines 8–10, buffer blocks are planned. In lines 11–19, the Lagrangian relaxation technique is invoked to reshape the floorplan. In line 20, unsatisfied nets are updated for the refined floorplan. In line 21, the resulting floorplan is partitioned into supermodules.

Procedure: SupermodulePartitioning($\Gamma, N, A_{max}, M_{max}$)
Input: Γ —A floorplan;
 N —A set of multi-terminal nets;
 A_{max} —Area threshold of a supermodule;
 M_{max} —Maximum # of modules in a supermodule;
Output: Γ —A partitioned floorplan;
1 RemoveTransitiveEdges(G_H, G_V);
2 **foreach** ($i \in G_H$) **do**
3 $M_i \leftarrow \{i\}$;
4 **while** ($\exists(i, j) \in G_H$ or $(i, j) \in G_V$ such that
5 $|M_i| + |M_j| < M_{max}$ and $\text{area}(M_i) + \text{area}(M_j) < A_{max}$) **do**
6 **if** (nets between M_i and M_j satisfy timing) **do**
7 $i \leftarrow i + j$;
8 $M_i \leftarrow M_i + M_j$;
9 Update(G_H, G_V);
10 $\Gamma \leftarrow \Gamma - \{j\}$.

Fig. 5. Supermodule partitioning procedure.

Procedure:
BufferBlockPlanning($\Gamma, N, A_{max}, M_{max}, B, \Psi$)
Input: Γ —Sequence pair of an intermediate floorplan;
 N —A set of multi-terminal nets;
 A_{max} —Area threshold of a supermodule;
 M_{max} —Maximum # of modules in a supermodule;
Output: B —A set of buffer blocks in Γ .
 Ψ —Unsatisfied nets.
1 (G_H, G_V) \leftarrow ConstraintGraph(Γ);
2 RemoveTransitiveEdges(G_H, G_V);
3 $B \leftarrow \Psi \leftarrow \emptyset$;
4 **for** $i = 1$ **to** n **do**
5 $T_i \leftarrow$ ConstructTree(i);
6 **if** (Unsatisfied(i)=1) **then**
7 $\Psi \leftarrow \Psi + \{i\}$;
8 **foreach** ($i \in N - \Psi$) **do**
9 AssignBuffer(i);
10 (G_H, G_V, B) \leftarrow GenerateBufferBlock();
11 (λ, μ, γ) \leftarrow InitializeMultiplier();
12 $k = 1$;
13 **repeat**
14 LRS_FindWidth();
15 LRS_FindBufferDistance();
16 (λ, μ, γ) \leftarrow AdjustMultiplier();
17 (λ, μ, γ) \leftarrow ProjectMultiplier();
18 $k \leftarrow k + 1$;
19 **until** converge
20 $\Psi \leftarrow$ CheckTiming(N);
21 $\Gamma \leftarrow$ SupermodulePartitioning($\Gamma, N, A_{max}, M_{max}$)

Fig. 6. Buffer-block planning procedure.

V. SIMULTANEOUS FLOORPLANNING AND BUFFER-BLOCK PLANNING (FBP)

In this section, we shall present our simultaneous FBP algorithm for the FBP problem. The FBP algorithm is based on simulated annealing and provides a mechanism to refine the floorplan. After perturbing the floorplan, FBP invokes the buffer-block planning procedure to plan buffers.

A. Solution Perturbation

A feasible nonslicing floorplan, without overlapping modules, can be represented by a sequence pair. We adopt the following four operations to perturb a sequence pair to another.

- Op1: Exchange two modules in the first sequence.
- Op2: Exchange two modules in both sequences.
- Op3: Rotate a module.
- Op4: Relax a supermodule.

Op1 swaps two modules in the first sequence only. Op2 swaps two modules in both sequences. Op3 rotates a module; eight orientations (with pin considerations) are configured for each module. Op4 relaxes a supermodule (decluster some modules in a supermodule). We perturb a solution with the guidance of the current solution. Hence, with a probability adjusted by temperature and the solution quality, the related modules of the unsatisfied nets are chosen as candidates for perturbation.

B. Cost Function

As given in Section II, the objective of the FBP problem is to find a floorplan with planned buffer blocks such that all timing requirements are satisfied and the area growth is minimized. Hence, a floorplan Γ is evaluated by its cost combined by area and timing as follows.

$$\text{cost}(\Gamma) = \text{area}(\Gamma) + \beta \sum_{1 \leq i \leq n} [D_{\text{net}}^i - D_{\text{req}}^i]^+$$

where β is a user specified parameter, N is the set of n nets, D_{net}^i is the delay of net i after buffer-block planning, D_{req}^i is the timing requirement of net i , and $[x]^+$ denotes the positive part of x , i.e., $[x]^+ = \max(0, x)$.

The first part of cost is the area consumed by the floorplan, including currently existing buffer blocks. The second part of the cost reflects the timing penalty paid for unsatisfied nets. The multiplier β means the area equivalent of time. In experiments, β is set to balance the area cost and timing penalty. The simulated annealing process gradually minimizes the cost.

C. Annealing Schedule

The annealing schedule controls the acceptance rate of uphill moves, neighboring solutions with higher costs. The initial temperature is set as $\Delta_{\text{avg}} / \ln(p)$, where Δ_{avg} is the average cost change of a random sequence of moves, and p is the initial probability of accepting uphill moves. In the beginning, the temperature is high; hence, p is initially set very close to 1. After each iteration, the temperature is reduced by a factor $\rho < 1$. The annealing process ends up when the temperature cools down below ε .

D. Overall Algorithm

The simulated annealing process begins from a random feasible floorplan Γ . Buffer blocks are accordingly planned as described in Section IV. FBP then perturbs the floorplan using the aforementioned four operations. After each move, buffer blocks are planned according to the new floorplan. The process terminates when the solution is frozen, the temperature is too low, or the runtime is too long.

Fig. 7 summarizes the FBP algorithm. In line 1, the initial floorplan is extracted from the benchmark circuits. In lines 5–31, FBP perturbs the floorplan from one to another until any of the conditions given in line 31 is satisfied.

VI. EXPERIMENTAL RESULTS

We implemented the FBP algorithm in the C language on a 166-MHz Sun UltraSPARC I workstation. The parameters used in the experiments are based on 0.18- μm technology (see Table I). Note that this set of parameters were also used in [5].

The statistics of benchmarks are outlined in Table II. It should be noted that, as presented earlier, our approach can handle

Algorithm: FBP(M, N, J, Γ, B, Ψ)
Input: M —A set of modules;
 N —A set of multi-terminal nets;
 J —# of moves per iteration.
Output: Γ —A sequence pair of the resulting floorplan;
 B —A set of buffer blocks in Γ .
 Ψ —Unsatisfied nets.

```

1   $\Gamma \leftarrow$  initial floorplan;
2   $(B, \Psi) \leftarrow$  BufferBlockPlanning( $\Gamma, N$ );
3   $Best\Gamma \leftarrow \Gamma$ ;  $BestB \leftarrow B$ ;  $Best\Psi \leftarrow \Psi$ ;
4   $\tau \leftarrow \frac{\Delta_{avg}}{\ln(p)}$ ;  $move \leftarrow uphill \leftarrow 0$ ;
5  do
6       $move \leftarrow uphill \leftarrow reject \leftarrow 0$ ;
7      do
8          SelectOperation(Op,  $\Psi, \tau$ );
9          Case Op:
10             Op1: Select two modules  $i, j$ ;
11                  $\Gamma' \leftarrow$  SwapFirst( $\Gamma, i, j$ );
12             Op2: Select two modules  $i, j$ ;
13                  $\Gamma' \leftarrow$  SwapBoth( $\Gamma, i, j$ );
14             Op3: Select one module  $i$ ;
15                  $\Gamma' \leftarrow$  Rotate( $\Gamma, i$ );
16             Op4: Select one supermodule  $i$ ;
17                  $\Gamma' \leftarrow$  Relax( $\Gamma, i$ );
18             Endcase
19              $move \leftarrow move + 1$ ;
20              $(B', \Psi') \leftarrow$  BufferBlockPlanning( $\Gamma', N$ );
21              $\Delta cost \leftarrow cost(\Gamma') - cost(\Gamma)$ ;
22             if ( $\Delta cost \leq 0$ ) or ( $Random < e^{-\frac{\Delta cost}{\tau}}$ ) then
23                 if ( $\Delta cost > 0$ ) then
24                      $uphill \leftarrow uphill + 1$ ;
25                      $\Gamma \leftarrow \Gamma'$ ;  $B \leftarrow B'$ ;  $\Psi \leftarrow \Psi'$ ;
26                 if ( $cost(\Gamma') < cost(Best\Gamma)$ ) then
27                      $Best\Gamma \leftarrow \Gamma'$ ;  $BestB \leftarrow B$ ;  $Best\Psi \leftarrow \Psi$ ;
28                 else  $reject \leftarrow reject + 1$ ;
29             while ( $(uphill \leq \frac{J}{2})$  and ( $move \leq J$ ));
30              $\tau \leftarrow \rho\tau$ ;
31             while ( $(\frac{reject}{move} \leq 0.95)$  and ( $\tau > \epsilon$ ) and ( $\text{!Out.Of.Time}$ ));
32              $\Gamma \leftarrow Best\Gamma$ ;  $B \leftarrow BestB$ ;  $\Psi \leftarrow Best\Psi$ 

```

Fig. 7. Simulated annealing for simultaneous FBP (the FBP algorithm).

TABLE II
STATISTICS OF BENCHMARKS

Circuit	# Modules	# Nets	# 2-terminal nets
apte	9	97	172
xerox	10	203	455
hp	11	83	226
ami33	33	123	363
ami49	49	408	545
playout	62	2506	2150

multiterminal nets directly. For a comparative study, however, we used the two-terminal nets obtained in [5] by splitting from multiterminal nets; the timing requirements are also generated by [5] from $1.05\text{--}1.20D_{opt}$. The experiments of [11] are based on different parameters and delay bounds (randomly generated within the same interval $1.05\text{--}1.20D_{opt}$), so we listed the results of the RBP algorithm in [11] only for the reader's reference. The experimental results are summarized in Table III. The second column shows the number of nets meeting timing requirements (# nets meet) and that of total nets in a circuit (Tot. # nets). The third column gives the percentages of nets meeting the timing constraints. Column 4 lists the number of buffers inserted (# buffers). Column 5 gives the percentages of extra areas over the given floorplans for buffer insertion. We

TABLE III
RESULTS OF BBP, FBP, AND RBP. THE EXPERIMENTS OF RBP ARE BASED ON DIFFERENT PARAMETERS AND DELAY BOUNDS (RANDOMLY GENERATED WITHIN THE SAME INTERVAL $1.05\text{--}1.20D_{opt}$), SO WE LISTED THE RESULTS OF THE RBP ALGORITHM FOR THE READER'S REFERENCE

Circuit Algorithm	# nets meet / Tot. # nets	% nets meet timing	# buffers	Extra area (%)
apte				
BBP	102 / 172	59.3	185	0.69
FBP	112 / 172	65.1	23	1.10
RBP	122 / 172	70.9	176	1.44
xerox				
BBP	260 / 455	57.1	399	1.38
FBP	389 / 455	85.5	184	0.00
RBP	368 / 455	80.8	354	1.24
hp				
BBP	131 / 226	58.0	280	1.24
FBP	196 / 226	86.7	37	0.00
RBP	185 / 226	81.9	258	1.03
ami33				
BBP	305 / 363	84.0	667	1.36
FBP	325 / 363	89.5	214	0.00
RBP	326 / 363	89.8	243	1.44
ami49				
BBP	412 / 545	75.6	946	0.78
FBP	513 / 545	94.1	280	0.00
RBP	497 / 545	91.2	287	1.04
playout				
BBP	1533 / 2150	71.3	4263	0.84
FBP	2055 / 2150	95.6	896	0.56
RBP	2053 / 2150	95.5	1090	1.32
Summary				
BBP	-	62.6	1123	1.05
FBP	-	86.1	272	0.28
RBP	-	85.0	401	1.25

compared with BBP [5]. In [5], BBP plans buffer blocks during postfloorplanning for two-terminal nets in a given slicing floorplan. (Note that FBP can handle multiterminal nets and general floorplans.) For fair comparison, FBP adopts buffer-block planning for two-terminal nets. In addition, FBP converts the given slicing floorplan into the corresponding sequence pair representation before processing. Runtime comparisons are not shown in this table because FBP not only planned buffer blocks but also refined floorplans. Further, BBP and FBP ran on different machines. For these benchmarks, the running times of FBP ranged from 1 min for the smallest circuit apte to about 35 min for the largest circuit playout. The results show that our method of integrating buffer-block planning into floorplanning can significantly improve the interconnect delay and reduce the number of buffers needed. FBP achieves an average success rate of 86.1% of nets meeting timing constraints, insert only 272 buffers on average, and consumes an average extra area of only 0.28% over the given floorplan, compared with the average success rate of 62.6%, 1123 buffers, and extra area of 1.05% resulted from BBP.

VII. CONCLUDING REMARKS

In this paper, we have addressed the issue of simultaneous FBP for interconnect optimization at the floorplanning stage. Experimental results have shown that our method can significantly improve the interconnect delay and reduce the number of buffers needed. For simultaneous FBP, besides interconnect delay, routing congestion and crosstalk could also be investigated in the future.

ACKNOWLEDGMENT

The authors would like to thank Prof. J. Cong, Dr. T. Kong, and Prof. D. Z. Pan for providing the benchmark circuits and their detailed explanations on the data. Thanks also go to the anonymous reviewers for their very constructive comments.

REFERENCES

- [1] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proc. 34th Design Automation Conf.*, June 1997, pp. 588–593.
- [2] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger, "Prim-Dijkstra tradeoffs for improved performance-driven routing tree design," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 890–896, July 1995.
- [3] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. G. Villarrubia, "A practical methodology for early buffer and wire resource allocation," in *Proc. 38th ACM/IEEE Design Automation Conf.*, June 2001, pp. 189–194.
- [4] H.-M. Chen, H. Zhou, F. Y. Young, D. F. Wong, H. H. Yang, and N. Sherwani, "Integrated floorplanning and interconnect planning," in *Dig. Tech. Papers 1999 IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1999, pp. 354–357.
- [5] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Dig. Tech. Papers 1999 IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1999, <http://cadlab.cs.ucla.edu/pan/publications/iccad99.ps>, pp. 358–363. revised version.
- [6] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.
- [7] J.-M. Ho, G. Vijaand, and C. K. Wong, "A new approach to the rectilinear Steiner tree problem," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 185–193, Feb. 1990.
- [8] M. Lai and D. F. Wong, "Maze routing with buffer insertion and wire-sizing," in *Proc. 37th ACM/IEEE Design Automation Conf.*, June 2000, pp. 374–378.
- [9] M. McInerney, K. Leeper, T. Hill, H. Chan, B. Basaran, and L. McQuiddy, "Methodology for repeater insertion management in the RTL layout, floorplan a fullchip timing databases of the Itanium™ microprocessor," in *Proc. ACM Int. Symp. Phys. Design*, Apr. 2000, pp. 99–104.
- [10] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1995, pp. 472–479.
- [11] P. Sarkar, V. Sundararaman, and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proc. ACM Int. Symp. Phys. Design*, Apr. 2000, pp. 186–191.
- [12] *National Technology Roadmap for Semiconductors*, 1997 ed: Semiconductor Industry Assoc..
- [13] *International Technology Roadmap for Semiconductors*, 1999 ed: Semiconductor Industry Assoc..
- [14] X. Tang and D. F. Wong, "Planning buffer locations by network flows," in *Proc. ACM Int. Symp. Phys. Design*, Apr. 2000, pp. 180–185.
- [15] W. L. Winston, *Operations Research: Applications and Algorithms*, 3rd ed. Toronto, Canada: Thomson, 1994.
- [16] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, 1986, pp. 101–107.
- [17] F. Y. Young, C. C. N. Chu, W. S. Luk, and Y. C. Wong, "Floorplan area minimization using Lagrangian relaxation," in *Proc. ACM Int. Symp. Phys. Design*, Apr. 2000, pp. 174–179.
- [18] H. Zhou, D. F. Wong, I.-M. Liu, and A. Aziz, "Simultaneous routing and buffer insertion with restrictions on buffer locations," in *Proc. 36th ACM/IEEE Design Automation Conf.*, June 1999, pp. 96–99.



Iris Hui-Ru Jiang received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1995 and 2002, respectively.

She is currently with VIA Technologies, Inc., Taipei, Taiwan. Her research interests focus on interconnect optimization in deep submicron technology.

Dr. Jiang is a Member of the ACM and ACM/SIGDA.



Yao-Wen Chang (S'94-M'96) received the B.S. degree from National Taiwan University, Taipei, in 1988 and the M.S. and the Ph.D. degrees from the University of Texas, Austin, in 1993 and 1996, respectively, all in computer science.

Currently, he is an Associate Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He was with the VLSI Design Group, IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994.

From 1996 to 2001, he was on the faculty of the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. His research interests lie in physical design automation, architectures, and systems for VLSI and combinatorial optimization.

Dr. Chang is a Member of IEEE Circuits and Systems Society, ACM, and ACM/SIGDA. He serves on the technical program committees of several international conferences on VLSI design automation, including ASP-DAC, ICCAD, ICCD, and APCCAS. He received a Best Paper Award at the 1995 IEEE International Conference on Computer Design (ICCD'95) for his work on FPGA routing, a reviewers' Best Paper nomination at the 2000 ACM/IEEE Design Automation Conference (DAC'2000) for his work on the B*-tree floorplan representation, and a Best Paper nomination at the 2002 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'2002) for his work on multilevel routing. He received an inaugural all-university Excellent Teaching Award from the Department of Computer and Information Science, National Chiao Tung University (ranked first in the Department) in 2000.



Jing-Yang Jou (S'82-M'83-SM'02) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana-Champaign, in 1979, 1983, and 1985, respectively.

He is currently a professor of the Department of Electronics Engineering at National Chiao Tung University, Hsinchu, Taiwan. He was previously with GTE Laboratories and Bell Laboratories. He has published more than 100 journal and conference papers. His research interests include behavioral and logic synthesis, VLSI designs and CAD for low power, design verification, and hardware/software codesign.

He is a Member of Tau Beta Pi. He served as the technical program chair of the Asia-Pacific Conference on Hardware Description Languages (APCHDL'97). He received the Distinguished Paper Award at the IEEE International Conference on Computer-Aided Design in 1990.



Kai-Yuan Chao received the B.S. degree in nuclear engineering from the National Tsing Hua University, Taipei, Taiwan, in 1986, the M.S. degree in medical engineering from the National Yang-Ming Medical College, Taipei, Taiwan, in 1988, and the M.S.E. and Ph.D. degrees in electrical and computer engineering from the University of Texas, Austin, in 1992 and 1995, respectively.

He presently manages floorplan and assembly design automation for major CPU development projects, including all Pentium 4 microprocessors, in the Desktop Platform Group, Intel Corporation, Hillsboro, OR, which he joined in 1995. He has published 19 technical papers and two book chapters in the research areas of VLSI/CAD, packaging, and radiology. His current research interests include architectural and design convergence, ECO methodology, and design collaboration.