Simultaneous Floorplanning and Buffer Block Planning *

Iris Hui-Ru Jiang[1], Yao-Wen Chang[2], Jing-Yang Jou[3], Kai-Yuan Chao[4]

[1]VIA Technologies Inc., Taipei 231, Taiwan

[2]Graduate Institute of Electronics Engineering & Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan

[3]Department of Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan

[4]Microprocessor Product Group, Intel Corporation, Hillsboro, OR 9712, USA

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. In this paper, we first address simultaneous floorplanning and buffer block planning (i.e., integrating buffer block planning into floorplanning) for interconnect optimization. Experimental results show that our method can significantly improve the interconnect delay and reduce the number of buffers needed.

I. Introduction

As revealed by the 1999 international technology roadmap for semiconductors [10], technology will soon shrink into below 0.1 micron, and the chip complexity will be over 200 million transistors. 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. The conventional design flow deals with interconnect optimization at the routing or the post-routing stage. When the amount of communication among modules rapidly increases, however, 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 [3]. 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 [4, 8, 11]. For a given floorplan, channels and dead spaces are used as buffer blocks, which accommodate buffers. Cong et al. first consider this issue in [4]. Sarkar et al. also consider routability and address the concept of independent feasible regions (the feasible regions of buffers for a net do not affect each other) in [8]. [4, 8] expand channels to accommodate more buffers if necessary. However, if the given floorplan is not good enough, channel expansion would result in much area overhead. Alpert et al. proposed buffer site methodology in [2], allocating buffers into empty silicon area inside macro blocks. However, current technology has not provided this kind of information; so buffer are still inserted outside macro blocks. Therefore, the existing buffer block planning is limited by the quality of a given floorplan.

In this paper, we first study simultaneous floorplanning and buffer block planning to conquer the weakness of the above. (In industry, this idea was considered for Intel Itanium microprocessor design [6].) Our method adopts the simulated annealing mechanism to refine a general 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.

*The work of Yao-Wen Chang was partially supported by the National Science Council of Taiwan ROC under Grant No. NSC-89-2215-E-009-117.

II. Problem Formulation

We define the Simultaneous Floorplanning and Buffer Block Planning (FBP) problem as follows.

- **Instance:** An initial floorplan, multi-terminal nets and their timing requirements, buffer library, technology file
- **Problem:** Find a floorplan with buffer block planning such that the area overhead is minimized subject to the timing requirement constraints.

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 [9]. These parameters were also used in [4, 8].

TABLE I
Parameters of 0.18 μm technology [9].

Parameter	Description (unit)	Value
$r_\square$	wire sheet resistance ($\Omega/\square$)	0.068
$\hat{r}$	wire unit-length resistance of 0.9 μm width ($\Omega/\mu m$)	0.075
c_a	wire sheet area capacitance ($fF/\mu m^2$)	0.06
c_f	wire fringing capacitance ($fF/\mu m$)	0.064
ω	wire width (μm)	0.9
$\hat{c}$	wire unit-length capacitance of 0.9 μm width ($fF/\mu m$)	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

III. Preliminaries

A. Sequence Pair Representation

We adopt the sequence pair representation [7] for a general floorplan. A sequence pair of a set of modules is a pair of sequences formed by module names. We can retrieve the topology relations and construct horizontal/vertical constraint graphs G_H/G_V as follows.

- **H-constraint:** If $(..a..b..,..a..b..)$, module b is on the right side of module a; there exists an edge (a, b) in G_H.

- **V-constraint:** If $(..a..b.., ..b..a..)$, module b is below module a; there exists an edge (a, b) in G_V

In G_H/G_V, each node is weighted as the module width/height. Two zero-weighted nodes s and t are also built. In addition, edges from s to zero-indegree nodes and from zero-outdegree nodes to t are added. 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).

B. Independent Feasible Region

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 [8].

Given a wire segment of length l with driver resistance R^D, load capacitance C^L, wire resistance per unit length $\hat{r}$, wire capacitance per unit length $\hat{c}$, buffer input resistance R_b, and buffer input capacitance C_b. Let $D^j_{net}(l_1, l_2, ..., l_n)$ 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 $D^j_{opt} = D^j_{net}(l^*_1, l^*_2, ..., l^*_n)$ are

$$l^*_i = \kappa_1 + (i-1)\kappa_2,\ i \in \{1, 2, ..., n\},$$

where

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

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

In [8], the independent feasible region F^j_i of width W^j_F for the i^{th} buffer of a net j is defined as

$$F^j_i = (l^*_i - \frac{W^j_F}{2}, l^*_i + \frac{W^j_F}{2}) \cap (0, l),$$

such that $(l_1, l_2, ..., l_n) \in F^j_1 \times F^j_2 \times ... \times F^j_n$ and $D^j_{net}(l_1, l_2, ..., l_n) \le D^j_{req}$, where D^j_{req} denotes the timing requirement associated with net j. Moreover, if $D^j_{req} \ge D^j_{opt}$, the width W^j_F of the independent feasible region for each buffer of net j is

$$W^j_F = 2\sqrt{\frac{D^j_{req} - D^j_{opt}}{\hat{r}\hat{c}(2n-1)}}.$$

C. Basic Buffer Block Planning

In this section, we propose the basic idea of our buffer block planning for two-terminal nets. (Multi-terminal nets will be considered later.) Based on the above formulae, the routing of a two-terminal net j 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 $l^*_i - W^j_F/2$ and $l^*_i + W^j_F/2$. A buffer block is a rectilinear region consisting of buffers, provided by dead spaces and/or channels. Each buffer is inserted into a buffer block with that its independent feasible region overlaps. If there are many choices, we first assign it to the one with the most overlapped area. After all buffers for all nets are allocated, the area of each buffer block is determined as the bounding area of inserted buffers. 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, record unsatisfied nets, assign buffer blocks (extended from the basic idea introduced in Section III), reshape the floorplan using the Lagrangian relaxation technique, update unsatisfied nets, 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 multi-terminal net. To construct a timing-aware routing tree for each net, we adopt the Prim-Dijkstra method proposed by [1], mixing Dijkstra's shortest path algorithm with Prim's minimum spanning tree one [5]. The generated tree tradeoffs between radius and wire length. The initial tree is then converted to a Steiner tree by removing overlapped edges. (Alternative tree construction approaches can also be used instead.)

B. Buffer Block Planning

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

Based on the longest path from the source to the sink terminals in the routing tree and the formulae described in Section B, we can check whether an optimally buffered routing tree can satisfy its timing requirement, i.e., $D^i_{opt} \le D^i_{req}$. We record these unsatisfied nets, and do not plan buffers for them, since the timing of these nets cannot be achieved. For the rest of nets, 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.

To preserve the topology of the routing tree, 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. 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.

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 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 [13] to facilitate area calculation, we add one dummy node labeled $m + b + 1$ to G_H and G_V. As mentioned in Section A, x_{m+b+1} (y_{m+b+1}) equals the width (height) of the packing. There are n multi-terminal nets. D^i_{req} denotes the timing requirement

of net i, and D^i_{net} 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.

$$\begin{array}{ll} Minimize & x_{m+b+1}y_{m+b+1} \\ Subject\ to & x_i + w_i \le x_j \qquad \forall (i,j) \in G_H, \\ & y_i + \frac{A_i}{w_i} \le y_j \qquad \forall (i,j) \in G_V, \\ & D^i_{net} \le D^i_{req}, \qquad \forall 1 \le i \le n, \\ & L_i \le w_i \le U_i \qquad \forall 1 \le i \le m+b. \end{array}$$

Because the objective function and the constraints are all posynomial [12], we can apply Lagrangian relaxation to solve the problem $\mathcal{PP}$ by introducing one non-negative Lagrange multiplier for each constraint. Therefore, the Lagrangian relaxation subproblem $\mathcal{LRS}$ is given by

$$\begin{array}{ll} Minimize & x_{m+b+1}y_{m+b+1} + \\ & \sum_{(i,j)\in G_H} \lambda_{i,j}(x_i + w_i - x_j) + \\ & \sum_{(i,j)\in G_V} \mu_{i,j}(y_i + \frac{A_i}{w_i} - y_j) + \\ & \sum_{1\le i\le n} \gamma_i (D^i_{net} - D^i_{req}) \\ Subject\ to & L_i \le w_i \le U_i, \ \ \forall 1 \le i \le m+b. \end{array}$$

The objective function of $\mathcal{LRS}$ is the Lagrangian function $L_{\lambda,\mu,\gamma}$. By Kuhn-Tucker conditions [12], we obtain the following theorems.

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

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

Theorem 2 *Let $(w_1, ..., w_{m+b})$ 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 distance between buffers of a net is constrained by

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

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

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 $\mathcal{PP}$.

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

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.

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

where $[x]^+ = \max(0, x)$ and $< \theta_k >$ is the step size sequence that satisfies $\lim_{k\to\infty} = 0$ and $\sum_{k=1}^{\infty} \theta_k = \infty$ (e.g., $\theta_k = \frac{1}{k}$). After the sub-gradient 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 into buffer block planning in the next iteration, and supermodules are considered as hard modules.

V. Simultaneous Floorplanning and Buffer Block Planning

Our simultaneous floorplanning and buffer planning (**FBP**) algorithm is based on simulated annealing and provides a mechanism to refine the floorplan.

A. Solution Perturbation

A feasible non-slicing 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.

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

A floorplan Γ is evaluated by its cost combined by area and timing as follows. $cost(\Gamma) = area(\Gamma) + \beta \sum_{1\le i\le n}[D^i_{net} - D^i_{req}]^+$, where β is a user specified parameter, N is the set of nets, D^i_{net} is the delay of net i after buffer block planning, D^i_{req} 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 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 to $\triangle_{avg}/ln(p)$, where $\triangle_{avg}$ 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 perturbes 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.

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 $0.18\ \mu m$ technology (see Table I). Note that this set of parameters were also used in [4].

It should be noted that, as presented earlier, our approach can handle multi-terminal nets directly. For comparative study, however, we used the two-terminal nets obtained in [4] by splitting from multi-terminal nets; the timing requirements are also generated by [4] from 1.05–$1.20D_{opt}$. *The experiments of [8] are based on different parameters and delay bounds* (randomly generated within the same interval 1.05–$1.20D_{opt}$), so we listed the results of the **RBP** algorithm in [8] for the reader's reference. The experimental results are summarized in Table II, including the number of nets meeting timing requirements (# nets meet) and that of total nets in a circuit (Tot. # nets), the percentages of nets meeting the timing constraints, the number of buffers inserted (#buffers), and the percentages of extra areas over the given floorplans for buffer insertion. For fair comparison with **BBP** in [4], **FBP** adopts buffer block planning for two-terminal nets and converts the given slicing floorplan into the corresponding sequence pair representation before processing. For these benchmarks, the running times of **FBP** ranged from 1 minute for **apte** to about 35 minutes for **playout**. The results show that our method 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**.

TABLE II
Comparison of **BBP**, **FBP** and **RBP**.

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

References

[1] 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. on Computer-Aided Design*, Volume 14, Issue 7, July 1995, pp. 890–896.

[2] C. J. Alpert, J. Hu, S. S. Sapatnekar, P. G. Villarrubia, "A Practical Methodology for Early Buffer and Wire Resource Allocation," *Proc. of 38th Design Automation Conf.*, June 2001, pp. 189–194.

[3] H.-M. Chen, H. Zhou, F.Y. Young, D.F. Wong, H.H. Yang and N. Sherwani, "Integrated Floorplanning and Interconnect Planning," *Proc. of 1999 Int'l Conf. on Computer Aided Design*, Nov. 1999, pp. 354–357.

[4] J. Cong, T. Kong and D.Z. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning," *Proc. of 1999 Int'l Conf. on Computer Aided Design*, Nov. 1999, pp. 358–363.

[5] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[6] M. Mclnerney, 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^{TM} Microprocessor," *Proc. of 2000 Int'l Symp. on Physical Design*, pp. 99–104, Apr. 2000.

[7] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-Packing-Based Module Placement," *Proc. of 1995 Int'l Conf. on Computer Aided Design*, Nov. 1995, pp. 472–479.

[8] P. Sarkar, V. Sundararaman and C.-K. Koh, "Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning," *Proc. of 2000 Int'l Symp. on Physical Design*, Apr. 2000, pp. 186–191.

[9] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, 1997 edition.

[10] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 1999 edition.

[11] X. Tang and D.F. Wong, "Planning Buffer Locations by Network Flows," *Proc. of 2000 Int'l Symp. on Physical Design*, Apr. 2000, pp. 180–185.

[12] W. L. Winston, *Operations Research: Applications and Algorithms*, 3rd ed., Thomson Publishing, 1994.

[13] F.Y. Young, C. C.N. Chu, W.S. Luk, and Y.C. Wong, "Floorplan Area Minimization using Lagrangian Relaxation," *Proc. of 2000 Int'l Symp. on Physical Design*, Apr. 2000, pp. 174–179.