Integrating Buffer Planning with Floorplanning for Simultaneous Multi-Objective Optimization*

Yi-Hui Cheng[1]*, and Yao-Wen Chang*[2]
Synopsys Inc., Taipei, Taiwan[1]
Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan[2]

Abstract

*As the process technology advances into the deep submicron era, interconnect plays a dominant role in determining circuit performance and signal integrity. Buffer insertion is one of the most effective and popular techniques to reduce interconnect delay and decouple coupling effects. It is traditionally applied to post-layout optimization. However, It is obviously infeasible to insert hundreds of thousands buffers during the post-layout stage when most routing regions are occupied. Therefore, it is desirable to incorporate buffer planning into floorplanning to ensure timing closure and design convergence. In this paper, we derive the formulae of feasible regions, and integrate buffer planning with floorplanning to optimize area, timing, noise, and congestion (routability) simultaneously. In particular, we treat each buffer block as a soft module and apply Lagrangian relaxation to optimize the floorplan area. Experimental results show that our method obtains an average success rate of 94.9% (86.4%) of nets meeting timing constraint alone (*both timing and noise constraints*) and consumes an average extra area of only 0.1% (0.2%) over the given floorplan.*

1 Introduction

As the process technology advances into the deep submicron era, interconnect plays a dominant role in determining circuit performance and signal integrity. Buffer insertion is one of the most effective and popular techniques to reduce interconnect delay and decouple coupling effects. It is shown that the delay of a signal wire is quadratically proportional to the wire length, and inserting buffers which break a wire into shorter wire segments makes the delay grow only linearly. A buffer can also be used to decouple the coupling capacitance (and thus improve signal integrity and delay) since it is a restoring device.

Traditionally, buffer insertion is performed during the post-layout or routing stage. As the SIA technology roadmap predicts, the number of buffers inserted for performance optimization will grow dramatically [14]. It is obviously infeasible to insert hundred of thousands buffers during the post-layout stage when most routing region is occupied by routing wires. Therefore, it is desirable to incorporate buffer planning into floorplanning to ensure timing closure and design convergence.

Researchers have proposed a few post-layout buffer-insertion algorithms in the literature. Alpert and Devgan [1] presented formulae for computing the optimal number and locations of buffers for timing optimization. Cong, Kong, and Pan in [5, 6] presented pioneering works on buffer block planning for interconnect-driven floorplanning. Given a slicing floorplan, they computed the *feasible regions* for inserting buffers to meet timing constraints, clustered buffers into blocks, and placed these buffer blocks into the dead spaces and channels in the slicing floorplan for timing optimization. The work by Sarkar Sundararaman, and Koh in [13] additionally considered the trade-off between routing congestion and circuit area. In [15], Tang and Wong applied a network-flow based algorithm to compute the maximum number of buffers that can be inserted into a feasible region. Dragan [9] presented the problem of buffer assignments for a given buffer block plan. More recently, Alpert et. al. [4] used a tile graph to distribute buffer sites throughout the layout for early buffer and wire resource allocation.

All the previous works on buffer planning do not consider crosstalk noise. Further, they work only *after* floorplanning when the related positions of modules have been fixed. In such a situation, the location and size of space for buffer insertion are also fixed. Since the deadspaces for buffer blocks are typically treated as unwanted cost during floorplanning, they are often avoided or minimized. As a result, the size and location of buffer blocks may not be suitable for later buffer insertion. Therefore, it is of significant importance to integrate buffer planning into floorplanning to fully utilize *useful* deadspaces for performance optimization. A previous work along this direction is due to Jiang et. al. [11] which planned buffers during floorplanning for timing optimization.

As mentioned earlier, due to the increasing design complexity, it may not be feasible to insert a large number of buffers for noise avoidance during the routing and post-layout stages. Therefore, there is a need to consider buffer planning earlier at the floorplanning stage when there is more flexibility to allocate silicon space for buffers to ensure timing closure and design convergence.

The rest of the paper is organized as follows. Section 2 gives some notations and models. Section 3 derives formulae for computing the feasible regions for inserting buffers. Section 4 presents our unified buffer planning and floorplanning algorithm. Section 5 shows the experimental results.

2 Preliminaries

In this section, we introduce the delay and the noise models applied in our work. Then, we define the problem of unified buffer planning and floorplanning with simultaneous delay, noise, area, and congestion consideration (*UBF* for short).

2.1 Delay Model

We model a buffer as a switch-level RC circuit and a wire segment as a π-model RC circuit. See Figure 1 for an illustration.

*This work was partially supported by the National Science Council of Taiwan under Grant No. NSC 91-2215-E-002-038.

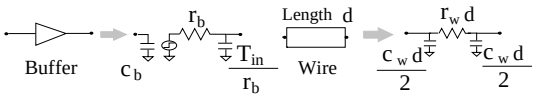

Figure 1: r_w and c_w are the sheet resistance and unit-length capacitance of a wire segment. c_b, r_b and T_{in} are the input capacitance, output resistance and intrinsic delay of a buffer.

As an example, for a two-pin net N of length d with driver resistance r_d and a load capacitance c_L. The delay of N is given as

$$\begin{aligned} D &= r_d(dc_w + c_L) + dr_w(\frac{dc_w}{2} + c_L) \\ &= r_d(dc_w + c_L) + \frac{r_w c_w d^2}{2} + dr_w c_L. \end{aligned} \quad (1)$$

Based on Equation (1) and the Elmore delay model, we can compute the delay of the net N with k buffers inserted as follows:

$$\begin{aligned} D &= r_d(d_0 c_w + c_b) + \frac{r_w c_w d_0^2}{2} + d_0 r_w c_b \\ &+ \sum_{i=1}^{k-1}(T_{in} + r_b(d_i c_w + c_b) + \frac{r_w c_w d_i^2}{2} + d_i r_w c_b) \\ &+ T_{in} + r_b(d_k c_w + c_L) + \frac{r_w c_w d_k^2}{2} + d_k r_w c_L, \end{aligned} \quad (2)$$

where d_i denotes the distance between buffer i and buffer $i+1$, as shown in Figure 2.

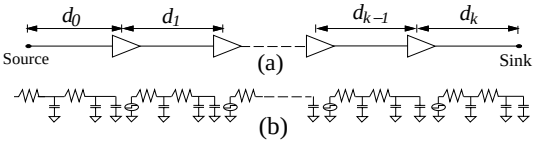

Figure 2: (a) A net with k buffers inserted. Here, d_i represents the distance between two adjacent buffers. (b) The corresponding RC model for buffers and wire segments.

2.2 Noise Model

Coupling noise between adjacent nets could lead to unexpected circuit behavior. Figure 3 shows the effect of coupling noise between two adjacent nets. The coupling capacitance (c_c) is proportional to the coupling length and inversely proportional to the distance between the aggressor and the victim nets. If the peak noise voltage (χ) is greater than the threshold voltage, it may cause the circuit to malfunction. Buffer insertion can be used to reduce noise effect since the buffer is a restoring logic gate that can prevent noise from being propagated from the input to the output of a buffer. In addition, inserting a buffer on a victim net will divide the net into two wire segments. As the coupling length is cut short, the coupling noise received by the victim net is reduced.

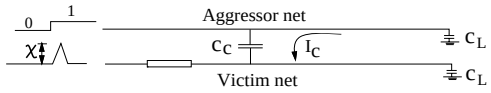

Figure 3: The noise model results from the coupling capacitance and crosstalk-induced current.

We adopt the noise model presented by Devgan [8]. The noise metric proposed in [8] has been shown to be the upper bound of noise for an RC circuit and specially useful for physical design based noise avoidance. Alpert, et. al. in [2] applied this noise metric to buffer insertion, and their results show that the metric is accurate and computationally efficient. As shown in [2], the coupling capacitance from an aggressor net to a victim net can be modeled as some fraction of the capacitance of the victim net. Consider a wire $e = (u, v)$ with t adjacent aggressor wire segments, where u and v are two nodes in a buffered tree. Let C_e and R_e denote the lumped capacitance and resistance for wire e. Its total induced noise current I_e can be computed by

$$I_e = C_e \sum_{i=1}^{t} \lambda_i \mu_i,$$

where λ_i is the ratio of coupling capacitance from an aggressor net i to the capacitance of the wire segment e, and μ_i is the slope (i.e., power supply voltage over input rise time) of the signal of an aggressor net i. Since the exact information about aggressor nets is not available before routing, we assume that μ_i is equal to μ for all aggressor nets i's. Thus, we have

$$I_e = C_e \Lambda \mu,$$

where $\Lambda = \sum_{i=1}^{t} \lambda_i$, and the amount of the additional noise of wire segment e is

$$\chi(e) = R_e \left(\frac{I_e}{2} + I_T(v) \right), \quad (3)$$

where $I_T(v)$ represents the total downstream current seen at node v.

2.3 Problem Formulation

The problem of unified buffer planning and floorplanning with simultaneous delay, noise, area, and congestion consideration (*UBF* for short) can be formally defined as follows: Given a set of modules and netlist, our objective is to find a feasible floorplan and the buffer plan to meet both timing and noise constraints and optimize area and wiring congestion at the same time.

3 Buffer Planning

First, we introduce the concept of buffer blocks and then show our theoretical results on the feasible region for each buffer under both noise and delay constraints.

3.1 Buffer Block

To place a buffer on the circuit, we use the concept of buffer block mentioned in [6]. In the following, we first define the buffer block. Then, we present a methodology to retrieve all existing buffer blocks efficiently.

3.1.1 Definition

Because buffers are made of transistors, the areas occupied by existing modules are considered obstacles during buffer allocation. The empty area in a floorplan is called *deadspace* which can be used to place buffers. In addition to deadspace, there are *channels* between adjacent modules. Using channels to place buffers must pay additional penalty since the channel needs to be expanded for inserting buffers, and thus the entire circuit area will increase. Both deadspace and channels are proper locations for buffer insertion. We use the term *buffer blocks* to represent those deadspace and channels that are occupied by buffers.

3.1.2 L-shaped Contour

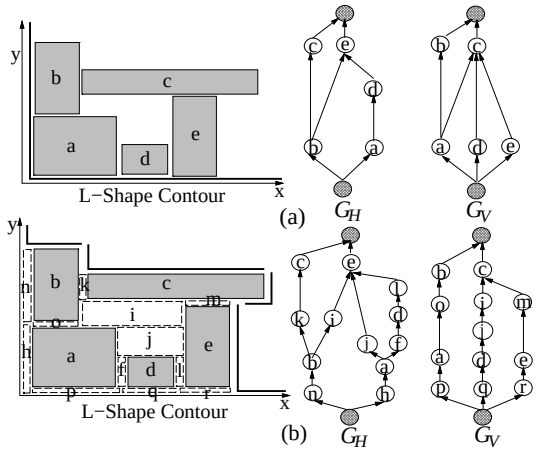

Figure 4: An example of buffer block retrieval with the L-shaped contour. (a) Initial contour in thick lines, and its corresponding G_H and G_V. $\{a, b, c, d, e\}$ represent hard modules. (b) The L-shaped contour, G_H and G_V after examining the 5 modules. $\{i, j\}$ are deadspace, $\{n, h, k, f, l\}$ are vertical channels, and $\{m, o, p, q, r\}$ are horizontal channels.

We develop a data structure called *L-shaped contour*, which not only can retrieve all information of deadspace and channels precisely, but also can rebuild the horizontal (vertical) constraint graph G_H (G_V) for *both modules and buffer blocks* at the same time. (G_H and G_V are used to model the relative positions of only modules in previous works.). Nevertheless, we need the constraint information of buffer blocks to refine the circuit area at the final stage. The L-shaped contour consists of a sequence of L-shaped segments. Initially the contour is just a big L representing the bottom-left boundary of the circuit, as shown in Figure 4(a). Then we examine all modules one by one according to the coordinates of their bottom-left corner. By examining each module, we can catch all the deadspaces on the left side of the module and locate all the channels along the boundaries of the module. See Figure 4(b) for an illustration. For example, the sorted order of those modules in the figure is $< a, b, c, d, e >$. While we are examining the last module e, we can retrieve the deadspace $\{i, j\}$, the vertical channel $\{l\}$, and the horizontal channel $\{m, r\}$. Meanwhile, the corresponding constraint graphs G_h and G_v are also updated during processing each module.

3.2 Feasible Region

Suppose that a net has met both the noise and delay constraints with k buffers inserted. Let b_i be the position of the i-th buffer ($1 \leq i \leq k$) on the net. With the buffer allocated in the Manhattan distance from the source of the net, we need to find how far the buffer can move away without violating any constraints while other buffers remain their positions. We call this range as *feasible region* of the buffer in which the buffer can move without violating any constraints.

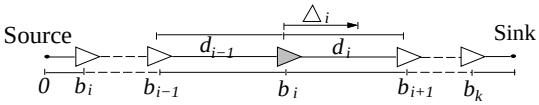

Figure 5: An illustration of moving the i-th buffer away by the distance Δ_i.

Because the position of a buffer can affect net delay significantly, we first use the following theorem to capture the effect of moving a buffer away from its original position on delay. Let the delay of a net with k buffers inserted be D.

Theorem 1 *If the i-th buffer moves away from its original position by the distance Δ_i. The delay D' is given by*

$$D' = D + r_w c_w \Delta_i^2 + r_w c_w (d_{i-1} - d_i) \Delta_i.$$

The new delay also needs to meet the delay constraint. Thus we have the following corollary.

Corollary 1.1 *If the i-th buffer moves away from its original position by the distance Δ_i. In order to meet the delay constraint T_{req}, Δ_i must satisfy the following requirements:*

$$-\sqrt{\frac{4T_{req} - 4D + \Delta_d^2}{4 r_w c_w}} - \frac{\Delta_d}{2} \leq \Delta_i$$

and

$$\sqrt{\frac{4T_{req} - 4D + \Delta_d^2}{4 r_w c_w}} - \frac{\Delta_d}{2} \geq \Delta_i,$$

where

$$\Delta_d = d_{i-1} - d_i.$$

With the above formulae, we can determine the feasible region of a buffer and update the delay increamentally if any buffer is moved away from its original position. To satisfy the delay constraints using buffer insertion, we apply those formulae proposed in [1] to decide how many buffers are needed and where the buffers are inserted. Let k_{opt} denote the optimal number of buffers needed to be inserted, x denote the range between the source and the first buffer, y denote the range between two adjacent buffers, and z denote the range between the last buffer and the sink. The optimal delay D_{opt} can be computed by the following equation:

$$\begin{aligned} D_{opt} &= r_d(c_b + c_w x) + \frac{r_w c_w x^2}{2} + r_w c_b x \\ &\quad + \sum_{i=1}^{k_{opt}-1} \left(T_{in} + r_b(c_b + c_w y) + \frac{r_w c_w y^2}{2} + r_w c_b y \right) \\ &\quad + T_{in} + r_b(c_b + c_w z) + \frac{r_w c_w z^2}{2} + r_w c_b z. \end{aligned}$$

As all buffers are placed at equal distance y, we can simplify Corollary 1.1 as follows:

$$-\sqrt{\frac{T_{req} - D}{r_w c_w}} \leq \Delta_i \leq \sqrt{\frac{T_{req} - D}{r_w c_w}}.$$

To compute the feasible region under delay constraints, we make the assumption that the feasible regions of buffers are independent, as [13] did.

If the peak noise voltage at the receiver is greater than the threshold voltage, it may cause the circuit to malfunction. We define the noise margin to keep the circuit from noise disturbance. In [2], Alpert et. al. used buffers to reduce the effect of noise due to wire coupling after the routing topology is decided. From their work, we know that a buffered net for delay optimization may not satisfy the noise constraint. Consider a wire $e = (u, v)$, where u and v are two nodes in a buffered tree. Let the length of the wire segment e be d. The maximum distance from v required for a buffer to satisfy the noise constraint is define as follows:

$$l_e \leq -\frac{r_b}{r_w} - \frac{I_T(v)}{i_w} + \sqrt{(\frac{r_b}{r_w})^2 + (\frac{I_T(v)}{i_w})^2 + \frac{2M_v}{i_w r_w}},$$

where $i_w = I_e/d$ denotes the current per unit length on net, and M_v is the noise margin at the downsteam point v of the wire segment.

Let the endpoints of the wire segment e be the i-th buffer and the $i+1$-th buffer. We can simplify the above inequality as follows:

$$l_i \leq -\frac{r_b}{r_w} + \sqrt{(\frac{r_b}{r_w})^2 + \frac{2M_{i+1}}{i_w r_w}},$$

where M_{i+1} denotes the tolerable noise margin of the downstream $i+1$-th buffer. The value of l_i is calculated from the position of the downstream buffer.

To handle delay and noise constraints simultaneously, we integrate all above formulae to find the feasible region for each buffer under noise and delay constraints. We present the integrated formula in the following theorem.

Theorem 2 *Given a buffered net with k buffers inserted, the feasible region Φ_i in which the i-th buffer can move from its original position without violating any timing and noise constraints is given by*

$$max(b_i - \Delta_i, b_{i+1} + l_i) \leq \Phi_i \leq min(b_i + \Delta_i, b_{i+1}).$$

Using Theorem 2, we can easily find the feasible region of each buffer to meet both delay and noise constraints. So far, we consider only the 1-D feasible distance from the source of a net for each feasible region and do not consider existing obstacles on the circuit. To obtain the 2-D feasible region, the feasible region Φ is evaluated as the Manhattan distance and the 2-D feasible region must be within the intersection of routing regions. See Figure 6 for an illustration.

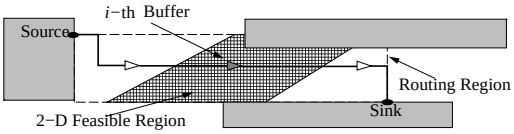

Figure 6: A 2-D feasible region for the i-th buffer

4 Unified Buffer Planning and Floorplanning

Figure 7 gives a main flow of our algorithm. First, we incorporate buffer insertion into simulated annealing based floorplanning. During simulated annealing, we apply buffer insertion to satisfy both delay and noise constraints. After expanding the circuit for the space of inserted buffers, we apply Lagrangian relaxation to refine the area of the floorplan.

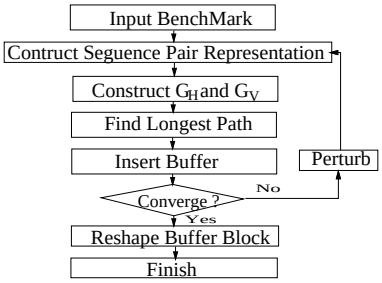

Figure 7: The flow of the unified buffer planning and floorplanning.

4.1 Simulated Annealing

Our unified buffer planning and floorplanning is based on simulated annealing and uses the sequence pair representation [12]. On the other hand, we use simulated annealing to search in the solution space for the optimal solution by perturbing the current solution. Perturbing a solution means that we can modify the sequence by exchanging two modules in either sequence or rotating some modules. The cost function of simulated annealing is defined as follows:

$$Cost = \alpha R_{fail} + \beta R_{area} + \gamma R_{congestion}.$$

The R_{fail} denotes the percentage of nets that fails to meet either the delay or noise constraint while the R_{area} evaluates the empty space and the expanded area due to buffer insertion.

Since each net must go through the buffer blocks that contain the buffers of the net, the locations of buffers will govern the route of each net. To handle the routing congestion earlier at the floorplanning stage, we introduce the third parameter $R_{congestion}$ to prevent the congestion at the routing stage. Let B_i represent the buffer block B_i, and C_h the set of the routing channels/regions in the chip. The congestion cost can be defined as follows:

$$R_{congestion} = \sum_i \{\frac{n_i}{w_i + h_i} \mid \forall B_i\} + \sum_{i \in C_h} \frac{\delta}{2^{c_i - d_i}}, \quad (4)$$

where w_i and h_i denote the width and height of B_i respectively, n_i denotes the number of nets whose buffers have been assigned to B_i, c_i and d_i denote the respective capacity and density of the routing channel/region i, and δ is a user-specified constant. The first term in Equation (4) considers the congestion through buffer blocks while the second captures the congestion of routing channels/regions. (Note that the global-route information can be approximated at this stage since we pre-route the nets for buffer planning.) For noise and delay constraints, we evaluate the R_{fail} after buffer insertion which will be discussed in the next section.

4.2 Buffer Insertion

The main objective of the buffer insertion is using buffers to meet both delay and noise constraints of nets, and locating those buffers on the circuit plane. First, we find where the buffers are available on the circuit plane. Since we use simulated annealing and the floorplan is updated in each iteration, we use the *L-shaped contour* mentioned in Section 3 to extract all the available space for buffers effectively. With the geometric information about all buffer blocks in the circuit, we can buffer each net to meet both constraints.

Before assigning buffers to buffer blocks, we need to know the number of buffers required, the locations of the buffers, and their feasible regions. For each net, we can compute the optimal number k_{opt} of buffers, the distance x between the source node and the first buffer, and the distance y between neighboring buffers by the formulae from [1]. With the delay formula, we can calculate D_{opt} from the source to the sink. If D_{opt} still cannot meet the delay constraint, we ignore the net and mark it infeasible.

Once the delay constraint is satisfied, we continue to insert buffers for noise constraints. From a sink to the source, we compute the maximum range l_i required for a buffer to meet the noise constraint for each buffer by the equations derived in Section 3. If the $(i-1)$-th buffer is within the range of l_i, the noise constraint is satisfied. Otherwise, we insert a new buffer at the position of $b_i + l_i$ to meet the noise constraints. While new buffers are inserted, we examine whether the delay still holds for the given delay constraint T_{req}. If the timing constraint cannot be satisfied due to the insertion of new buffers, we ignore the net and mark it infeasible.

By the theorems in Section 3, the feasible region of each buffer can be derived easily. With the feasible region of each buffer and the geometric information of buffer blocks, we can find a feasible buffer block for buffer insertion. If the buffer cannot find any feasible buffer block, we ignore the net and mark it failed.

A buffer may have multiple choices for feasible buffer blocks. Each buffer block within a buffer's feasible region is a feasible solution to meet both delay and noise constraints. To minimize the circuit area and channel expansion after buffer insertion, we choose the buffer block with the largest capacity. If all buffer blocks are full, we choose the one which results in the minimum expanded area. Once a buffer is assigned to a buffer block, the width and height of the buffer block should be updated accordingly. After assigning all buffers, we start to run the longest path algorithm on the constraint graphs to update the width and the height of the circuit respectively.

4.3 Reshaping Buffer Block

The buffers being inserted into channels or overflowed deadspace may cause the channels or deadspace to be expanded. Once the buffer block is expanded, the expansion may create new deadspace, relocate modules, and make the chip larger. To minimize the chip area, we may treat buffer blocks as soft module and reshape the buffer blocks based on Lagrangian relaxation [16].

Suppose that we have n modules in the circuit, including both hard modules and buffer blocks which are treated as soft modules. G_H and G_V keep the relative locations of the n modules. Let x_i, y_i, w_i and A_i denote the x-coordinate, y-coordinate, width, and area of module i respectively. Each module i is given a minimum width L_i and a maximum width U_i. In addition to the n module nodes, let node $n+1$ denote the sink in each constraint graph. After the longest path algorithm is applied on the constraint graphs, x_{n+1} (y_{n+1}) give the x-coordinate (y-coordinate) of the rightmost (topmost) boundary. As a result, the primal problem denoted by PP becomes

$$\begin{array}{ll} Minimize & x_{n+1}y_{n+1} \\ Subject\ to & x_i + w_i \leq x_j \quad \forall e(i,j) \in G_H, \\ & y_i + \frac{A_i}{w_i} \leq y_j \quad \forall e(i,j) \in G_V, \\ & L_i \leq w_i \leq U_i \quad \forall 1 \leq i \leq n, \end{array}$$

where $e(i,j)$ means that there is an edge from node i to node j in the constraint graph. Applying Lagrangian relaxation, we introduce nonnegative multipliers $\rho_{i,j}$ and $\xi_{i,j}$ for each edge. Then we obtain the Lagrangian relaxation subproblem $LRS(\rho, \xi)$ as follows.

$$\begin{array}{ll} Minimize & x_{n+1}y_{n+1} + \\ & \sum_{e(i,j) \in G_H} \rho_{i,j}(x_i + w_i - x_j) + \\ & \sum_{e(i,j) \in G_V} \xi_{i,j}(y_i + \frac{A_i}{w_i} - y_j) \\ Subject\ to & L_i \leq w_i \leq U_i \\ & \forall 1 \leq i \leq n. \end{array}$$

By the Kuhn-Tucker conditions, we can obtain the optimal conditions for those Lagrangian multiplier, ρ and ξ in $LRS(\rho, \xi)$. The optimality conditions are as follows:

$$\sum_{e(j,i) \in G_H} \rho_{j,i} = \sum_{e(i,j) \in G_H} \rho_{i,j}, \quad (5)$$

$$\sum_{e(j,i) \in G_V} \xi_{j,i} = \sum_{e(i,j) \in G_V} \xi_{i,j}, \quad (6)$$

$$y_{n+1} = \sum_{e(i,n+1) \in G_H} \rho_{i,n+1}, \quad (7)$$

$$x_{n+1} = \sum_{e(i,n+1) \in G_V} \xi_{i,n+1}. \quad (8)$$

If (ρ, ξ) satisfies the above conditions, the objective function F of $LRS(\rho, \xi)$ can be simplified as

$$F = \sum_{1 \leq i \leq n} \left(\left(\sum_{e(i,j) \in G_H} \rho_{i,j} \right) w_i + \left(\sum_{e(i,j) \in G_V} \xi_{i,j} \right) \frac{A_i}{w_i} \right) - \left(\sum_{e(i,n+1) \in G_H} \rho_{i,n+1} \right) \left(\sum_{e(i,n+1) \in G_V} \xi_{i,n+1} \right). \quad (9)$$

To get the optimal value of w_i, F is differentiated with respect to w_i. Then we

Parameter	Description (unit)	Value
r_w	sheet resistance of a net ($\Omega/\mu m$)	0.075
c_w	unit-length capacitance of a net ($fF/\mu m$)	0.118
T_{in}	intrinsic delay for a buffer (ps)	36.4
c_L	load capacitance (fF)	23.4
r_d	driver resistance (Ω)	180.0
c_b	input capacitance of a buffer (fF)	23.4
r_b	output resistance of a buffer (Ω)	180.0
M_v	the noise margin for a buffer or a sink v (V)	0.8

Table 1: Parameters of the 0.18μm technology in NTRS'97.

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

Table 2: Statistics of the MCNC benchmark circuits.

have the following equation:

$$w_i = \sqrt{\frac{A_i \sum_{e(i,j)\in G_V} \xi_{i,j}}{\sum_{e(i,j)\in G_H} \rho_{i,j}}}.$$

Recall that w_i must be within the range $[L_i, U_i]$. Thus, the optimal value of w_i becomes

$$w_i = min(U_i, max(L_i, \sqrt{\frac{A_i \sum_{e(i,j)\in G_V} \xi_{i,j}}{\sum_{e(i,j)\in G_H} \rho_{i,j}}})). \quad (10)$$

Let $Q(\rho, \xi)$ denote the optimal value of $LRS(\rho, \xi)$. The Lagrangian dual problem is to find a vector of Lagrangian multipliers such that the optimal solution of LRS is also the optimal solution of PP. We define the Lagrangian dual problem LDP as follows.

$$\begin{array}{ll} Maximize & Q(\rho, \xi) \\ Subject\ to & \rho \geq 0\ and\ \xi \geq 0. \end{array}$$

To find those optimal multipliers, we iteratively adjust multipliers by the subgradient optimization method to find the optimal width of each block. Starting from an arbitrary (ρ, ξ) under the optimality condition by step k, we move to a new vector (ρ', ξ') by following the subgradient direction:

$$\rho'_{i,j} = [\rho_{i,j} + \sigma_k(x_i + w_i - x_j)]^+ \quad (11)$$

$$\xi'_{i,j} = [\xi_{i,j} + \sigma_k(y_i + \frac{A_i}{w_i} - y_j)]^+ \quad (12)$$

where $[x]^+ = \max(0, x)$ and $< \sigma_k >$ is the step size sequence that satisfies $\lim_{k\to\infty} = 0$ and $\sum_{k=1}^{\infty} \sigma_k = \infty$. After the subgradient optimization, Lagrangian multipliers may be changed. The new multipliers need to be projected back to the nearest point meeting the optimality conditions.

Because PP is a convex problem, we can know that if (ρ, ξ) is the optimal solution to LDP, the optimal solution of $LRS(\rho, \xi)$ will also optimize PP. By iteratively adjusting multipliers and Equation 10, we can obtain the optimal width of each block and thus minimize the entire chip area.

5 Experimental Results

The UBF algorithm was implemented in the C++ language on a 450 MHz SUN Ultra 60 workstation and experimented on the six MCNC benchmark circuits used in [5, 6, 13, 11]. See Table 2 for the statistics of the circuits. In addition to UBF$_{d\&n}$ which considers both noise and delay constraints, we also implemented UBF$_d$ which considers only delay constraint for fair comparison. We shall focus on the comparisons of delay, noise, and area.

The parameters for interconnects and buffers were based on the $0.18\mu m$ technology given in the NTRS'97 roadmap [14] and were used in [5, 6, 11, 13] (see Table 1 for the parameters). All parameters used are the same as [2, 5, 6, 11]. We adopted the rise time for an aggressor net as 0.25 ns, the power supply voltage as 1.8 V, and noise margin as 0.8 V. Same as [5, 6, 11], all nets were 2-pin connections and the power/ground nets were excluded. Also, we used the same delay constraints provided by the authors of [5, 6], which were randomly generated in $[1.05T_{opt}, 1.20T_{opt}]$.

We compared our results with the previous works BBP/FR [5, 6] and FBP [11]. (We also list the results of IFR [13] for reader's reference. However, it should be noted that the delay constraints used for IFR are slightly different from those used for BBP/FR and our UBF.) As mentioned earlier, BBP/FR and IFR apply buffer insertion *after* floorplanning. After floorplanning, all modules are placed and thus all available rooms for buffers are also limited. Unlike those works, FBP and our UBF performed buffer insertion *during* floorplanning to fully utilize deadspace. Besides, our UBF can consider not only delay but also noise while BBP/FR, IFR, and UBF consider only delay.

Table 3 gives the number and the percentage of nets meeting the given constraints, the percentages of areas increased after buffer insertion, computed by $\frac{(expanded\ chip\ area - original\ chip\ area)}{original\ chip\ area}$, and the total numbers of buffers inserted to satisfy the given constraints. In all experiments, BBP/FR, IFR, FBP, and UBF$_d$ consider only delay constraints while UBF$_{d\&n}$ considers *both delay and noise constraints*. As shown in these tables, UBF$_d$ (UBF$_{d\&n}$) achieves an average success rate of 94.9%(86.4%) of nets meeting timing (both timing and noise) constraints and consumes an average extra area of only 0.10% (0.22%), compared with the average success rate of 72.8% (86.1%) meeting timing constraints and extra area of 1.03% (0.28%) resulted from BBP/FR (FBP). As for IFR, it obtains average success rate of 84.8% nets meeting timing constraints and consumes an average extra area of 1.25%. The experimental results show that UBF$_d$ and UBF$_{d\&n}$ perform better than BBP/FR, FBP, and IFR in delay and area.

#Net Meet / Meet Ratio (%)					
Circuit	BBP	IFR*	FBP	UBF$_d$	UBF$_{d\&n}$
apte	132/76.7	122/70.9	112/65.1	155/90.1	137/79.7
xerox	304/66.8	368/80.9	389/85.5	454/99.8	441/96.9
hp	154/68.1	185/81.9	196/86.7	222/98.2	188/83.2
ami33	302/83.2	326/89.8	325/89.5	345/95.0	329/90.6
ami49	398/73.0	497/91.2	513/94.1	540/99.1	506/92.8
playout	1478/68.7	2020/93.9	2055/95.6	1878/87.4	1618/75.3
Avg.	72.8	84.8	86.1	94.9	86.4
Area Expansion Ratio (%)					
Circuit	BBP	IFR*	FBP	UBF$_d$	UBF$_{d\&n}$
apte	1.44	1.44	1.10	0.16	0.35
xerox	1.39	1.24	0.00	0.00	0.31
hp	1.05	1.03	0.00	0.34	0.34
ami33	0.93	1.44	0.00	0.00	0.00
ami49	0.65	1.04	0.00	0.00	0.06
playout	0.71	1.32	0.56	0.12	0.23
Avg.	1.03	1.25	0.28	0.10	0.22
#Buffers Inserted					
Circuit	BBP	IFR*	FBP	UBF$_d$	UBF$_{d\&n}$
apte	262	176	23	129	248
xerox	519	354	184	324	527
hp	301	258	37	244	281
ami33	703	243	214	366	314
ami49	949	287	280	410	879
playout	4262	1090	896	1234	3348
Avg.	1242	446	272	451	932

Table 3: Experimental results for the number of nets (#Net Meet) and the percentage of nets (Meet Ratio) meeting the delay constraint (also noise constraint for UBF$_{d\&n}$) and the corresponding area expansion ratio and the number of buffers inserted.

The overhead of UBF$_{d\&n}$ for meeting the additional noise constraints is consuming more area for buffer insertion. Because there are delay and noise constraints with UBF$_{d\&n}$, the feasible region for each buffer could be smaller than that for delay optimization alone. According to the experimental results, nevertheless, UBF$_{d\&n}$ even achieved better delay and area than the previous works in spite of the more stringent constraints. Further, even consuming more buffers than IFR and FBP, UBF still achieved a smaller average percentage of area expansion. The two facts revealed that UBF can utilize the deadspace and routing channels/regions more effectively during floorplanning. The running time of UBF$_d$ (UBF$_{d\&n}$) averages about 8 (11) minutes and ranges from 29 seconds (40 seconds) for the smallest circuit **apte** to 33 (45) minutes for the largest circuit **playout**, compared to the running time of FBP which ranges from 1 minute for **apte** to 35 minutes for **playout** on a 166 MHz SUN Ultra Sparc machine. Both UBF and FBP required much longer running times than BBP/FR and IFR because UBF and FBP not only plan buffer locations but also perform floorplanning by using simulated annealing while BBP/FR and IFR only plan buffer locations.

References

[1] C. J. Alpert and A. Devgan, "Wire Segmenting for Improved buffer insertion," *Proc. of DAC*, pp. 588–593, June 1997.

[2] C. J. Alpert and A. Devgan, S. T. Quay, "Buffer Insertion for Noise and Delay Optimization," *IEEE Trans. CAD*, vol. 18, No. 11, pp. 1633–1645, Nov. 1999.

[3] 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 CAD*, vol. 14, pp. 890–896, 1995.

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

[5] J. Cong, T. Kong and Z. D. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning," *Proc. of ICCAD*, pp. 358–363, Nov. 1999.

[6] J. Cong, T. Kong and Z. D. Pan, "Buffer Block Planning for Interconnect Planning and Predicition," *IEEE Trans. VLSI Systems*, 2001.

[7] J. Cong, and D. Z. Pan, "Wire Width Planning for Interconnect performance Optimization," *IEEE Trans. CAD*, vol. 21, pp. 319–329, Mar. 2002.

[8] A. Devgan, "Efficient coupled noise estimation for on-chip interconnects," *Proc. of ICCAD*, pp. 147–151, Nov. 1997.

[9] F. Dragan, A. Kahng, I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering using an available buffer block plan," *Proc. of ICCAD*, pp. 104–109, Nov. 2000.

[10] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide band amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55–63, 1948.

[11] I. H.-R. Jiang, Y.-W. Chang, J.-Y. Jou, and K.-Y. Chao, "Simultaneous floorplanning and buffer block planning," *Proc. of ASPDAC*, pp. 431–434, January 2003.

[12] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair" *IEEE Trans. CAD*, vol. 15, pp. 1518–1524, 1996.

[13] P. Sarkar, V. Sundararaman and C.K. Koh, "Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning," *Proc. of ISPD*, pp. 186–191, April 2000.

[14] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, 1997 Edition.

[15] X. Tang and D.F. Wong "Planning Buffer Locations by Network Flows " *Proc. of ISPD*, pp. 180–185, April 2000.

[16] F. Y. Young, Chris C. N. Chu, W. S. Luk, and Y. C. Wong "Handling soft modules in general nonslicing floorplan using Lagrangian relaxation" *IEEE Trans. CAD*, Vol. 20, pp. 687–692, 2001.