Routability Driven Floorplanner with Buffer Block Planning

Chiu Wing Sham
Department of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, N.T. Hong Kong
cwsham@cse.cuhk.edu.hk

Evangeline F. Y. Young
Department of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, N.T. Hong Kong
fyyoung@cse.cuhk.edu.hk

ABSTRACT

In traditional floorplanners, area minimization is an important issue. However, due to the recent advances in VLSI technology, the number of transistors in a design are increasing rapidly and so are their switching speeds. This has increased the importance of interconnect delay and routability in the overall performance of a circuit. We should consider interconnect planning, buffer planning and routability as early as possible. In this paper, we study and implement a routability-driven floorplanner with congestion estimation and buffer planning. Our method is based on a simulated annealing approach that is divided into two phases: the area optimization phase and the congestion optimization phase. In the area optimization phase, modules are roughly placed according to the total area and wirelength. In the congestion optimization phase, a floorplan will be evaluated by its area, wirelength, congestion and routability. We assume that every buffer should be inserted at a flexible interval from each other for long enough wires and probabilistic analysis is performed to compute the congestion information taken into accounts the constraints in buffer locations. Our approach is able to reduce the average number of wires at the congested areas and allow more feasible insertions of buffers to satisfy the delay constraints without having much penalty in increasing the area of the floorplan.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and Routing*

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

1.1 Motivations

ISPD'02, April 7-10, 2002, San Diege, California, USA

Floorplanning plays an important role in physical design of VLSI circuits. It plans the shapes and locations of the modules on a chip, and the result of which will greatly affect the overall performance of the final circuit. In the past, area minimization is the major concern in floorplan design. Advances in the deep sub-micron technology have brought many changes and challenges to this. As technology continues to scale down, the sizes of the transistors are getting smaller and a significant portion of the circuit delay is coming from interconnects. In some advanced systems today, as much as 80% of the clock cycle is consumed by interconnects [2]. Area minimization is less important while routability and delay has become the major concern in floorplanning and many other designing steps.

Traditional floorplanners have not paid enough attention to interconnect optimization. This results in a large expansion in area, or even in an unroutable design failing to achieve timing closure after detailed routing. Buffer insertion is one of the most popular and effective techniques to achieve timing closure. It was projected that over 700K repeaters will be inserted on a single chip in the 70 nm technology [3]. In the current practice, buffers are inserted after routing. However buffers also take up silicon resources (40× to 200× minimum inverter size) and cannot be inserted wherever we want. A good planning of the module positions during the floorplanning stage so that buffers can be inserted wherever needed in the later routing stages will be very useful. Besides, buffers contribute delay and area, and their locations should be carefully planned.

1.2 Previous Works

There are several previous works addressing the interconnect issues in floorplan design. In the paper [2], a floorplan is divided into grids and congestion is estimated at each grid, assuming that each wire is routed in either L-shape or Z-shape. They use a simple and fast congestion model but buffer insertion is not considered. Cong et al. define in their paper [4] the term *feasible region* of a net, that is, the largest polygon in which a buffer can be inserted such that the timing constraint can be satisfied. The locations of these feasible regions can be computed and buffers are clustered into blocks in these feasible regions along the channel areas. Sarkar et al. [9] add into the notion of independence to feasible regions so that the feasible regions of different buffers on a net can be computed independently. They also try to improve routability by considering congestion in their objective function. Tang and Wong [10] propose an optimal algorithm to assign buffers to buffer blocks assuming that only one buffer is needed per net. In the paper [11], a re-

alistic global router is used to evaluate congestion of each candidate placement solution. Dragan et al. [5] use a multi-commodity flow-based approach to allocate buffers to some pre-existing buffer blocks such that the required upper and lower bounds on buffer intervals can be satisfied as much as possible. Alpert et al. [1] make use of tile graph and dynamic programming to perform buffer block planning. They propose that buffers should be allowed to be inserted inside macro blocks and their method will distribute buffer sites throughout the layout. Finally, Lou et al. [8] apply probabilistic analysis to estimate congestion and routability, and they show that their estimations correlate well with post-route congestion. However their congestion model does not take into account buffer insertions.

1.3 Our Contributions

In order to address the interconnect issues in floorplan design, we propose to use a probabilistic method to estimation congestion and routability with buffer insertions. Probabilistic analysis can address the problem in a more global way and can thus give a more accurate estimation on the interconnect information. In our model, we assume that buffers are constrained to be inserted for long enough wires such that the distance between adjacent buffers is lying within a range $[low, up]$ given by the user. We call this the *variable interval buffer insertion constraint* [5, 1]. For example, according to [7], global repeater rules for a high-end microprocessor design in the 0.25 μm CMOS technology require repeaters at intervals of at most 4500 μm, and we can model this situation by assigning the low and up appropriately. In our floorplanner, we will divide a floorplan into a 2-dimensional grid structure and will estimate congestion at each grid subjecting to the given buffer insertion constraint. Notice that the congestion at each location is dependent on the routes of the wires while the route of a wire is, in turn, dependent on the availability of buffer resources along the route. This availability of buffer resources can be estimated from the amount of empty space at each location and the number of possible buffer insertions at that location. We compute the congestion information assuming that every route is equally likely to be used as long as the buffer insertion constraint can be satisfied. The computation can be performed efficiently by using dynamic programming and a table lookup approach. Besides, we have used a two stage simulated annealing method to speed up the whole floorplanning process. In order to verify the usefulness of our method, we have implemented a simple global router for testing and the experimental results can demonstrate the effectiveness and efficiency of our floorplanner.

This paper is divided into eight sections. In section 2, an overview of our floorplanner will be given. In section 3-7, the ideas and the implementation details including buffer planning, congestion estimation and the two-phases simulated annealing process will be described and explained. Finally, experimental results and comparisons between our floorplanner and a traditional floorplanner will be shown in section 8.

2. OVERVIEW OF OUR FLOORPLANNER

In this section, we will give a brief overview for our routability driven floorplanner. We assume that wires are routed over-the-cell and multi-bend routing is used. We divide a floorplan into grids and the size of the deadspace in each grid is computed for estimation of the amount of buffer resources. Given a set of m nets and a set of n modules where each module M_i has an area A_i, we want to obtain a non-overlap packing of these modules such that the area of the packing, the interconnect cost and the congestion cost are small, every net satisfies its buffer insertion constraint and every module satisfies its area and aspect ratio constraint.

In our floorplanner, given a candidate floorplan solution, we will first estimate the buffer usage of each wire k at each grid, then we can estimate the total buffer usage at each grid, $b\_usage(x, y)$. We assume that buffers should be inserted at flexible intervals from each other for long enough wires. By making use of $b\_usage(x, y)$ and the estimation on the amount of buffer resources $b\_space(x, y)$ ($b\_space(x, y)$ is estimated from the size of deadspace at grid (x, y)), we can estimate the probability of successful buffer insertion $b\_success(x, y)$ at each grid. We can then make use of this information to estimate the congestion information, $weight(x, y)$. In order to improve the routability of the design, our approach will try to reduce the congestion of the interconnect at different locations of the layout and to ensure that the delay of a net can satisfy its delay constraint by buffer insertions.

3. CONGESTION MODEL

In order to improve the routability of the floorplan solution, a congestion model is used to estimate the congestion information. By using a good congestion model, the intermediate floorplan solutions can be evaluated accurately according to their congestion and routability. The paper [8] shows that their probabilistic congestion model correlates well to post-route results but their work does not consider buffer locations. This motivates us to use a probabilistic model in our floorplanner.

A commonly used congestion model is shown in figure 1. We can see that the probability for each possible route of a wire k is equal. However, if the probabilities of successful buffer insertion (dependent on the amount of buffer resources) are different at different grids, the probabilities of some routes will be higher if they pass through those locations with higher probabilities of successful buffer insertion.

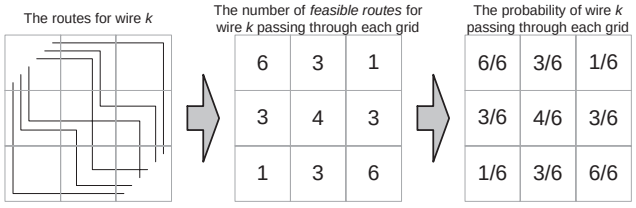

Figure 1: A Commonly Used Congestion Model

In our approach, we try to address the buffer insertion issue in a new congestion model. In this model, adjacent buffers are required to be inserted at a distance $x \in [low, up]$ from each other where low and up are the lower and upper bound of the buffer insertion constraint given by the users. In figure 2, we assume that both the upper and lower bound of the buffer insertion constraint are equal to two, i.e., buffers should be inserted at an interval of two grid unit

length from each other. We can see that the probabilities of the routes b, c, d and e with successful buffer insertions are only 0.5. Therefore, the total number of *feasible routes* for this wire is reduced from six to four where the number of *feasible routes* is equal to the summation of the probabilities of all possible routes satisfying the buffer insertion constraint. As a result, the probability of wire k passing through each grid will be different from that computed in figure 1. The probabilities of some routes become higher if they pass through the locations with higher probabilities of successful buffer insertion. In order to compute the congestion information using this model, we need estimates of $b\_usage(x, y)$ and $b\_success(x, y)$ at each grid (x, y). In this section, we will show the computation of the congestion information, assuming that $b\_success(x, y)$ is given for each grid (x, y). We will show how we can estimate $b\_success(x, y)$ in section 4.

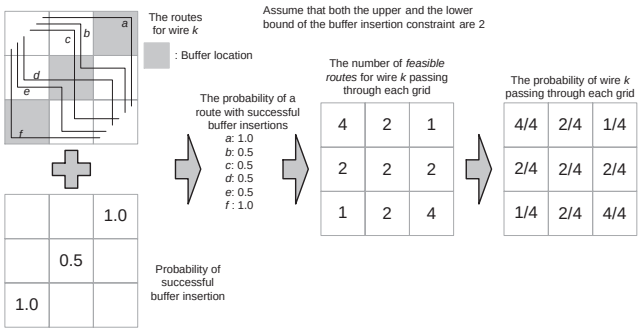

Figure 2: Our Congestion Model

3.1 Probabilistic Model with Variable Interval Buffer Insertion Constraint

Given the information on buffer resources at each grid, we want to calculate the probability $r\_success(l)$ that a route l can be routed successfully from the source to the sink without violating any buffer insertion constraint. In the example of figure 3, there are only two feasible ways of buffer insertion for each route because the wirelength is six and the upper bound and lower bound of the buffer insertion constraint are three and two respectively. The probability of the first feasible way with all buffers inserted successfully is equal to $b\_success(x_0 + 2, y_0) \times b\_success(x_0 + 4, y_0)$, while that of the second one is equal to $b\_success(x_0 + 3, y_0)$. Given the probabilities of successful buffer insertion, the probability of route 1 satisfying the buffer insertion constraint, $r\_success(1)$, can be computed as $\frac{0.2+0.5}{2} = 0.35$. Similarly, we can calculate $r\_success(l)$ for all the other routes l. Thus, the total number of *feasible routes* of a wire k is equal to $\sum_{l \in L_k} r\_success(l)$ where L_k is the set of all routes for wire k, and the number of *feasible routes* passing through the grid (x, y) is equal to $\sum_{l \in L_k(x,y)} r\_success(l)$ where $L_k(x, y)$ is the set of all routes for wire k passing through the grid at (x, y). For the example in figure 3, $\sum_{l \in L_k} r\_success(l)$ is equal to $3 \times 0.25 + 2 \times 0.35 + 0.1 = 1.55$. Consider the grid at (x_0+1, y_0), route 1 ,2, 3, 4 and 5 will pass through this grid, so the total number of *feasible routes* passing through this grid is equal to $2 \times 0.25 + 2 \times 0.35 + 0.1$, i.e., 1.3. Consider the grid at $(x_0, y_0 + 1)$, only route 6 will pass through this grid.

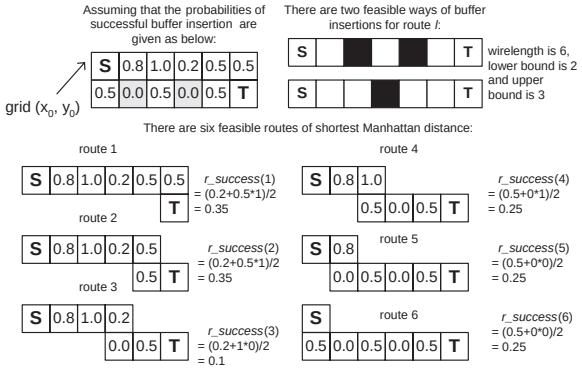

Figure 3: Example of Computing $r\_success(l)$

Thus, the total number of *feasible routes* passing through this grid is equal to $r\_success(6)$, i.e., 0.25.

Consequently, the congestion information $F(x, y, k)$ at grid (x, y) for wire k can be calculated as:

$$F(x, y, k) = \frac{\sum_{l \in L_k(x,y)} r\_success(l)}{\sum_{l \in L_k} r\_success(l)},$$

where $F(x, y, k)$ is the probability of wire k passing through grid (x, y). For example, we can compute $F(x_0 + 1, y_0, k) = \frac{1.3}{1.55}$ and $F(x_0, y_0 + 1, k) = \frac{0.25}{1.55}$. We can see that $F(x_0 + 1, y_0, k) + F(x_0, y_0 + 1, k)$ is equal to one because the routes of wire k must either pass through the grid at (x_0+1, y_0) or at (x_0, y_0+1). Finally, the average number of wires passing through a grid at (x, y) is computed as:

$$weight(x, y) = \sum_{all\ wire\ k} F(x, y, k)$$

In order to calculate $F(x, y, k)$ efficiently, we use dynamic programming. At the beginning, we are given the probability of successful buffer insertion at each grid. We will then create an array $inf[0...up]$ at each grid with size equal one plus the upper bound of the buffer insertion constraint, up. Assuming that wire k starts from grid (x_0, y_0) and ends at grid (x_t, y_t), we will initialize $g_0(x_0, y_0, k).inf[0]$ to one and the remaining values of the array to zero. The value of $g_0(x, y, k).inf[i]$ represents the total number of *feasible routes* from the source to (x, y) such that the last buffer is inserted at a grid of distance i units before (x, y). The values of the array at each grid can be computed dynamically row by row accordingly to the following recursive equation:

$$\begin{aligned} v_1 &= g_0(x-1, y, k).inf[i-1] + g_0(x, y-1, k).inf[i-1] \\ v_2 &= \sum_{i=low}^{up} g_0(x, y, k).inf[i] \times b\_success(x, y) \end{aligned}$$

$$g_0(x, y, k).inf[i] = \begin{cases} v_1 & \text{for } 1 \le i \le up \\ v_2 & \text{for } i = 0 \end{cases}$$

Consequently, the value $g_0(x_t, y_t, k).inf[0]$ can be obtained that represents the total number of *feasible routes* for wire k from the source (x_0, y_0) to the sink (x_t, y_t). Then, we will repeat the same steps from the sink (x_t, y_t) to the source (x_0, y_0) to calculate $g_1(x, y, k).inf[i]$s. The total number of feasible routes passing through the grid at (x, y) and a buffer is inserted at (x, y) can be computed as:

$$F_1(x,y,k) = \frac{g_0(x,y,k).inf[0] \times g_1(x,y,k).inf[0]}{b\_success(x,y)}.$$

In addition, the total number of *feasible routes* passing through the grid (x, y) and no buffer is inserted at (x, y) can be computed as:

$$F_0(x,y,k) = \sum_{i,j \neq 0; low \leq i+j \leq up} g_0(x,y,k).inf[i] \times g_1(x,y,k).inf[j]$$

Finally, the probability that wire k will pass through the grid at (x, y) can be computed as:

$$F(x,y,k) = (F_0(x,y,k) + F_1(x,y,k))/g_0(x_t,y_t,k).inf[0]$$

3.2 Time Complexity

According to this congestion estimation model, we need to scan the array at each grid from the source to the destination for each net. Therefore, the time complexity is $O(m \times k \times up)$ where k is number of grids scanned for each net, m is number of nets and up is the upper bound of the buffer insertion constraint. If the floorplan is divided into $K \times K$ grids, we need to scan at most K^2 grids for each net. Moreover the upper bound in the buffer insertion constraint is also bounded by the length of the wires. (It is bounded by $2K$). Therefore, the time complexity of this congestion estimation is only $O(m \times K^3)$.

4. BUFFER PLANNING

In this section, we will show how we can estimate the amount of buffer usage $b\_usage(x, y)$ and the probability of successful buffer insertion $b\_success(x, y)$ at each grid (x, y). Such information will be used in the calculation of the congestion information discussed in section 3.1.

4.1 Estimation of Buffer Usage

According to the variable interval buffer insertion constraint, buffers can be inserted at flexible locations as long as adjacent buffers are at a distance $x \in [low, up]$ from each other. There are thus several feasible ways of buffer insertion satisfying the constraint for each route. The probability that a route l of wire k will insert a buffer at grid (x, y), $b\_insert(x, y, l, k)$ should be calculated in order to estimate the amount of buffer usage $b\_usage(x, y)$. Given a possible route l of wire k with source S and sink T and the buffer insertion constraint $[up, low]$, the total number of feasible ways of buffer insertions, *total*, can be obtained. We assume that all feasible ways of buffer insertions are equally likely to be used in the routing stage. The probability that a grid (x, y) is required to have a buffer inserted for a route l of wire k can be calculated as:

$$b\_insert(x,y,l,k) = \frac{N(dist)}{total},$$

where *dist* is the Manhattan distance of the grid (x, y) from the source S, *total* is the total number of feasible ways of buffer insertions for l, and $N(dist)$ is the number of feasible ways of buffer insertions that will insert a buffer at a grid of distance *dist* from the source S. In the example of figure 4, there are four feasible ways of buffer insertions satisfying the buffer insertion constraint, so *total* is equal to four. Consider the grid (x_0+4, y_0), its distance from the source S is equal to four, so *dist* is four. There is only one feasible way of buffer insertions that will insert a buffer at a grid of distance four from the source, so $N(4)$ is equal to one and the probability that grid $(x_0 + 4, y_0)$ is required to have a buffer inserted can be computed as:

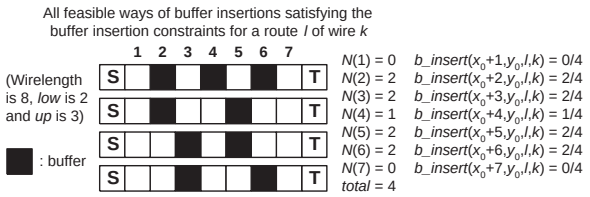

Figure 4: Example of Calculating the Probability of Successful Buffer Insertion at each Grid

$$b\_insert(x_0 + 4, y_0, l, k) = n(4)/4 = 0.25$$

We can compute these $b\_insert(x, y, l, k)$ from $cnt(D, up, low)$ where $cnt(D, up, low)$ is the number of feasible ways of buffer insertion from a grid A to another grid B such that D is the unit grid distance between A and B, and up and low are the bounds of the buffer insertion constraint. The probability of successful buffer insertion at a grid (x, y) of a distance *dist* from the source can be calculated as:

$$b\_insert(x,y,l,k) = \frac{cnt(dist,up,low) \times cnt(length-dist,up,low)}{cnt(length,up,low)},$$

where *length* is the distance from the source S to the sink T of route l, $cnt(dist, up, low)$ is the number of feasible ways of buffer insertion from the source to grid (x, y), $cnt(length - dist, up, low)$ is the number of feasible ways of buffer insertion from the grid (x, y) to the sink and $cnt(length, up, low)$ is total the number of feasible ways of buffer insertions from the source to the sink.

In order to compute the probability of buffer insertion, $b\_insert(x, y, l, k)$ at each grid (x, y) for every route l and wire k efficiently, two methods based on dynamic programming are used. In both methods, $cnt(D, up, low)$ can be calculated, saved and reused. They are the forward recursive method and the backward recursive method as described below.

The forward recursive method works by searching and counting the number of feasible ways of buffer insertions satisfying the buffer insertion constraint. For each successful way of buffer insertion found, the number accumulated will be increased by one. Finally, the total number of feasible ways of buffer insertions between two points of distance D will be obtained. In general, the function will be called recursively until all feasible ways of buffer insertions are found. The formula is shown below:

$$cnt(D,up,low) = \begin{cases} \sum_{i=low}^{up} g_1(D,i,up,low) & \text{if } D \geq low \\ 0 & \text{if } D < low, \end{cases}$$

$$\text{where } g_1(D,i,up,low) = \begin{cases} cnt(D-i,up,low) & \text{if } D \neq i \\ 1 & \text{if } D = i \end{cases}$$

The backward recursive is a faster method but it has a limitation that the lower bound in the buffer insertion constraint must be one. It counts the number of infeasible ways of buffer insertions by looking at the first position where a violation occurs. Finally, it computes the number of infeasible ways of buffer insertions, $fail\_total$. Since the total number of ways of buffer insertions is 2^{D-1}, the total number of feasible ways of buffer insertion satisfying the buffer insertion constraint, *total*, can be calculated as $2^{D-1} - fail\_total$. The general formula is shown below:

$$cnt(D,up,1) = \begin{cases} 0 & \text{if } D = 0 \\ g_2(D,up) & \text{if } D > 0, \end{cases}$$

where $g_2(D, up) = 2^{D-1} - \sum_{i=0}^{D-up-1} 2^{D-i-up-1} \times cnt(i, up, 1)$

Assuming that the time required for each recursive call is similar for the two methods, we find that the backward recursive method is faster than the forward one when $up > \frac{D}{5}$.

We can combine the two methods to compute the total number of feasible ways of buffer insertions satisfying the buffer constraints efficiently. Given the distance ($length$) of a route l of a wire k running from a source S to a sink T and the distance ($dist$) from the source to a grid at (x, y). We can compute the total number of feasible ways of buffer insertions from S to T such that a buffer is inserted at (x, y) as:

$$cnt(dist, up, low) \times cnt(length - dist, up, low)$$

Therefore, the probability that a buffer is required at grid (x, y) for route l of wire k is:

$$b\_insert(x, y, l, k) = \frac{cnt(dist,up,low) * cnt(length-dist,up,low)}{cnt(length,up,low)}$$

Note that these values can be computed, saved and reused for the wires of the same length.

4.2 Estimation of Buffer Resources

In order to obtain the congestion information, we need to compute the probability of successful buffer insertion at each grid. We have shown in the above section the computation of the probability that a route l of wire k will insert a buffer at a grid (x, y), i.e., $b\_insert(x, y, l, k)$, the total number of buffer insertions at grid (x, y), $b\_usage(x, y)$, can then be calculated as:

$$b\_usage(x, y) = \sum_{all\ wire\ k\ and\ all\ route\ l} b\_insert(x, y, l, k)$$

The probability of successful buffer insertion at grid (x, y), $b\_success(x, y)$, can be calculated as:

$$b\_success(x, y) = min(1, \frac{b\_space(x,y)}{b\_usage(x,y)}),$$

where $b\_space(x, y)$ is the number of buffers that can be inserted at grid (x, y) and is dependent on the amount of empty space in grid (x, y). From the equation, we can see that if the amount of empty space at grid (x, y) is large enough to accommodate for all possible buffer insertions, the value of $b\_success(x, y)$ is equal to one. Otherwise, it is equal to $b\_space(x, y)/b\_usage(x, y)$.

5. TWO-PHASES SIMULATED ANNEALING

In our design, the simulated annealing process is divided into two phases. They are the area optimization phase and the congestion optimization phase. In the area optimization phase, the congestion information is less meaningful because the locations of the modules are still far from their final position. Thus, the cost function used in this area optimization phase does not include the congestion cost and the buffer insertion cost. The cost function is shown below:

$$Cost_0 = Area + \alpha \times Wire,$$

where $Area$ is the area of the floorplan and $Wire$ is the total wirelength (it will be discussed in section 6) and α is the weight.

In the congestion optimization phase, the cost function is shown below.

$$Cost_1 = Area + \alpha \times Wire + \beta \times M\_weight,$$

where $Area$ is the area of the floorplan, $Wire$ is the total wirelength, M_weight is the average number of wires in the top ten percent most congested grids and α and β are the weights.

In the transitional period from the area optimization phase to the congestion optimization phase, we need to re-calculate the temperature because the order of magnitude of the cost might change a lot and the acceptance rate could drop or rise unexpectedly if we did not adjust the temperature. As a result, we need to use a new temperature to maintain the acceptance rate. To achieve this, we will first obtain the mean value of $\triangle Cost$ from a number of random walks using the old cost function, and obtain the mean value of $\triangle new\_Cost$ from a number of random walks using the new cost function. The new temperature $temp$ can then be computed as below:

$$temp = \frac{\triangle new\_Cost}{\triangle Cost} \times old\_temp$$

By using $temp$, the acceptance rate can be maintained and the change in the cost function can be performed smoothly.

6. WIRELENGTH ESTIMATION

In the floorplanning stage, the positions of the I/O pins in the modules are not yet fixed and the modules will usually cover a number of grids. Therefore, the assumption on the locations of the I/O pins will affect the results significantly. Consequently, we need to estimate reasonably the positions of the I/O pins in order to compute the interconnect cost and congestion cost accurately.

In order to distribute the I/O pins into the grids appropriately, intersection-to-intersection method is used. Consider a net connecting two modules A and B, we will first draw a line from the center of one module to another. The two intersecting points will be found on the edges of the modules and the I/O pins will be placed at the grids containing the intersecting points.

This is an appropriate method for our floorplanner since the estimated positions of the I/O pins will be similar to those in the final design and the buffer locations can be estimated more accurately.

7. MULTI-PIN NETS HANDLING

In order to handle multi-pin nets, we need to decompose a multi-pin net into a set of two-pin nets. There are several methods to decompose a multi-pin net into two-pin nets such as using minimum spanning tree (MST), rectilinear steiner tree (RST). MST runs faster but it may over-estimate the congestion because of the overlapping net segments. However, this conservative estimation will not affect the resultant packing significantly because the total length of an MST can be reduced at most by 6% to 9% only by removing all the overlapping net segments to obtain a corresponding RST [6]. Since the runtime of an RST algorithm is usually much slower than that of an MST algorithm, MST is a better choice for estimation purposes in the early floorplanning stage. As a result, we apply MST to handle multi-pin nets in our floorplanner.

8. EXPERIMENTAL RESULTS

We have implemented three floorplanners, a traditional floorplanner $F1$ based on simulated annealing without considering congestion and buffer planning, a routability-driven

Cases	*ami*33			*ami*49			*playout*		
Floorplanners	$F1$	$F2$	$F3$	$F1$	$F2$	$F3$	$F1$	$F2$	$F3$
[3] Area ($k\mu m^2$)	1292.19	1381.78	1311.27	39712.15	39634.84	39727.66	1000.46	984.03	993.22
[3] Wirelength ($k\mu m$)	22.08	23.22	21.35	412.05	477.35	403.95	285.41	323.44	276.02
[1] Congestion	3.13	1.74	2.06	0.18	0.17	0.16	23.18	22.54	22.20
No. of unroutable wires	78.88	67.88	72.55	125.25	119.12	112.75	490.38	465.75	456.88
[2] Runtime (s)	127.71	3339.92	678.45	151.56	3660.31	789.46	494.45	32100.50	3498.23

[1] Average number of nets per $10^3\mu m^2$ for the top 10% most congested grids. [2] Experiments are performed using Pentium IV 1.2GHz with 512Mb memory. [3] Before enlargement.

Table 1: Comparisons between $F1$, $F2$ and $F3$

floorplanner $F2$ based on simulated annealing using the probabilistic model for buffer planning with variable interval buffer insertion constraint, and another routability-driven floorplanner $F3$ that is similar to $F2$ except that two-phases simulated annealing is used. The data sets used in the experiments are *ami*33, *ami*49 and *playout*. The buffer insertion constraint $[low, up]$ used are [3,6], [2,4] and [3,6] respectively in terms of grid lengths (approximately equal to $[2100\mu m, 4200\mu m]$). The value of low and up are computed as below:

$$low = \frac{\sqrt{\frac{4*(R_b*C_b+D_b)}{R_o*C_o}}}{(2*10*g\_unit)}, up = \frac{\sqrt{\frac{4*(R_b*C_b+D_b)}{R_o*C_o}}}{(10*g\_unit)}$$

g_unit is the length of a grid and the others are the physical parameters in the $0.18\mu m$ technology. Notice that the sizes of the modules are enlarged for demonstration of the effect of buffer block planning. We have implemented a simple global router to evaluate the performance of the floorplanners. In the router, the nets will be routed one after another and a wire is unroutable when it cannot be routed form the source to the sink in the shortest Manhattan distance and satisfying the buffer and congestion (we assume that there is limitation on the number of wires in each grid) constraints.

Comparing $F1$ and $F3$ in table 1, we can see that the differences between $F1$ and $F3$ on area are very small but $F3$ can reduce the total wirelength, congestion and number of unroutable wires significantly. It demonstrates the effectiveness of our probabilistic congestion model in reducing the interconnect cost in floorplan design. Notice that the area penalty of $F3$ is very small and the runtime is only double that of $F1$.

If we compare between $F2$ and $F3$, we can see that the performance of $F2$ and $F3$ are similar except that the runtime of $F3$ is much smaller. This demonstrates that the two-phases simulated annealing approach can reduce the runtime of the whole floorplanning process significantly without any degradation in performance.

9. CONCLUSION

In this paper, we present a new congestion model to compute the congestion information taking into account buffer planning with variable interval buffer insertion constraint. The estimations are based on the supply and demand analysis of routing and buffer resources, where the supply is determined by the floorplan solution (the amount of empty space) and the demand is determined by the interconnect structure. Computations of the congestion information in our model are quite complicate but they can be performed efficiently by dynamic programming. Experimental results show that our floorplanner can reduce the interconnect cost efficiently without much penalty in area. In addition, the runtime can be improved significantly without degradation in performance by the two-phases simulated annealing process.

10. REFERENCES

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

[2] H. M. Chen, H. Zhou, F. Y. Young, and D. Wong. Integrated floorplanning and interconnect planning. In *Proc. Int. Conf. On CAD*, pages 354–357, 1999.

[3] J. Cong. Challenges and opportunities for design innovations in nanometer technologies. In *Frontiers in Semiconductor Research: A Collection of SRC Working Papers*, 1997.

[4] J. Cong, T. Kong, and D. Pan. Buffer block planning for interconnect driven floorplanning. In *CAD 1999. Digest of Technical Papers. 1999 IEEE/ACM Int. Conf.*, pages 358–363, 1999.

[5] F. F. Dragan, A. B. Kahng, S. Muddu, and A. Zelikovsky. Provably good global buffering using an available buffer block plan. In *Proc. Int. Conf. On CAD*, 2000.

[6] J. M. Ho, G. Vijayan, and C. K. Wong. A new approach to the rectilinear steiner tree problem. In *26th ACM/IEEE DAC*, 1989.

[7] A. B. Kahng, S. Muddu, E. Sarto, and R. Sharma. Interconnect tuning strategies for high performance ics. In *Pro. DATE*, 1998.

[8] J. Lou, S. Krishnamoorthy, and H. S. Sheng. Estimating routing congestion using probabilistic analysis. In *Int. Sym. on Physical Design, ACM*, pages 112–117, April 2001.

[9] P. Sarkar, V. Sundararaman, and C. K. Koh. Routability-driven repeater block planning for interconnect-centric floorplanning. In *ISPD 2000*, 2000.

[10] X. P. Tang and D. Wong. Planning buffer locations by network flows. In *Intl. Symp. Physical Design*, pages 186–191, 2000.

[11] M. Wang and M. Sarrafzadeh. Modeling and minimization of routing congestion. In *Proc. of Asian-Pacific DAC*, pages 185–190, 2000.