Dynamic Global Buffer Planning Optimization Based on Detail Block Locating and Congestion Analysis

Yuchun Ma[1], Xianlong Hong[1], Sheqin Dong[1], Song Chen[1], Yici Cai[1], C.K.Cheng[2], Jun Gu[3]
[1] *Department of Computer Science & Technology, Tsinghua University, Beijing, China,100088* [2] *Department of Computer Science and Engineering, University of California,San Diego CA 92093-0114,USA*
[3] *Department of Computer Science, Science & Technology University of HongKong*
clara99@mails.tsinghua.edu.cn; hxl-dcs@tsinghua.edu.cn

ABSTRACT
By dividing the packing area into routing tiles, we can give the budget of the buffer insertion. And the detail locating of the blocks in their rooms can be implemented for each iterations during the annealing process to favor the later buffer planning. The buffer insertion will affect the possible routes as well the congestion of the packing. The congestion estimation in this paper takes the buffer insertion into account. So we devise a buffer planning algorithm to allocate the buffer into tiles with congestion information considered. The buffer allocation problem is formulated into a net flow problem and the buffer allocation can be handled as an integral part in the floorplanning process. Since there is more freedom for floorplan optimization, the floorplanning algorithm integrated with buffer planning can result in better performance and chip area.

Categories and Subject Descriptors
B.7.2 [**Integrated Circuits**]: Design Aids – *Placement and Routing*

General Terms
Algorithms, Performance, Design.

Keywords
Floorplanning, Routability, Congestion, Buffer Insertion.

1. INTRODUCTION
Due to the recent advances in VLSI technology, the number of transistors in a design is increasing rapidly and so are their switching speeds. This has increased the importance of the interconnect delay in the overall performance of a circuit. Many techniques are employed to reduce interconnect delay. Among them, buffer insertion has shown to be an effective approach to achieve timing closure. As transistor count and chip dimension get larger and larger, more and more buffers are expected to be needed for high performance. It was projected that over 700K buffers will be inserted on a single chip in the 70nm technology[1]. Since buffers are implemented by transistors, they cannot be placed over the existing circuit blocks. Placing a large number of buffers between circuit blocks could significantly impact the chip floorplan. Therefore, it is necessary to start buffer planning as early as possible. It is very useful that a good planning of the block positions can be obtained during the floorplanning stage so that buffers can be inserted wherever needed in the later routing stages.

1.1 Previous Works
There are several previous works addressing the interconnect issues in floorplanning design. Cong[2] define the term "feasible region" (FR) of a net, that is, the largest polygon in which a buffer can be inserted such that the timing constraint can be satisfied. Sarkar[3] added the notion of independence into feasible regions so that the feasible regions of different buffers on a net can be computed independently. These two papers gives the basic idea of Feasible Region, based on which they proposed the buffer planning algorithms. But both of their methods take complex scanning to obtain the feasible buffer insertion sites. Tang and Wong[4] propose an optimal algorithm based on net flow to assign buffers to buffer blocks assuming that only one buffer is needed per net. Alpert et al.[5] make use of tile graph and dynamic programming to perform buffer block planning, while they propose that buffers should be allowed to be inserted inside macro blocks. But all of these algorithms are based on fixed die placement and it is difficult to embed those methods into the iterations of the floorplanning process because of the complexity of those algorithms. Unfortunately, the fixed placement is likely to generate some timing-constraint violations which are beyond repair unless the topological relation between blocks can be changed. Sham[6] proposed a routability driven floorplanner while the buffer insertion are assumed to be inserted at a flexible interval from each other for long enough wires. Therefore, the buffer insertion is just estimated by probabilistic budget, while the buffer blocks are not allocated in Ref.[6]. Hence, to create a performance-feasible floorplan, a performance-driven floorplanner that simultaneously considers area and buffer block insertions is needed. Although many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2003, June 2-6, 2003, Anaheim, California, USA.
Copyright 2003 ACM 1-58113-688-9/03/0006…$5.00

This work is supported by the National Natural Science Foundation of China 60121120706 and National Natural Science Foundation of USA CCR-0096383, the National Foundation Research(973) Program of China G1998030403, the National Natural Science Foundation of China 60076016 and 863 Hi-Tech Research & Development Program of China 2002AA1Z1460, the National Doctoral Foundation of China 20020003008.

approaches have been proposed to make use of the dead space for buffer insertion, few of them optimize the buffer planning by changing the distribution of the dead-space in the placement.

1.2 Our Contributions

Previous methods of buffer planning are based on the fixed blocks. We have found that by changing the positions of the blocks within their rooms will not affect the total area and the topological relations, but the buffer insertions may be optimized. In our floorplanner, we will divide the packing area into routing tiles and we can give the budget of the buffer insertion at each tile. Therefore, we devise a novelty method of the detail process of locating the blocks in their rooms to favor the buffer insertions. The dead spaces in each tile can be computed.

The buffer insertion will influence the routing ways greatly. So we estimate the congestion information based on 2-bend routes with some blocked tiles considered. This method is very effective since we can use the routes matrix to figure out all the possible routes instead of computing all the routes by scanning the tiles. Taking advantage of the buffer budget and congestion estimation, the buffer planning algorithm is based on net flow method to optimize both the congestion and the buffer insertion. To speed up our algorithm, the simulated annealing process is divided into two phases: timing optimization phase and buffer insertion phase.

The rest of the paper is composed as follows: Sect.2 gives the overview of our floorplanner; Sect.3 gives the budget of the buffer insertion and the floorplanning algorithm to locate the circuit blocks in their rooms. The buffer planning algorithm and the two-phase annealing process are described in Sect.4 and Sect.5. The experimental results are shown in Sect.6. Finally, the conclusion is given.

2. OVERVIEW OF OUR FLOORPLANNER

In this paper, we concentrate on the buffer planning problem with the floorplanning. The insertion of buffers should be in the dead spaces between circuit blocks. We seek a floorplanning methodology to produce the optimal floorplan such that the

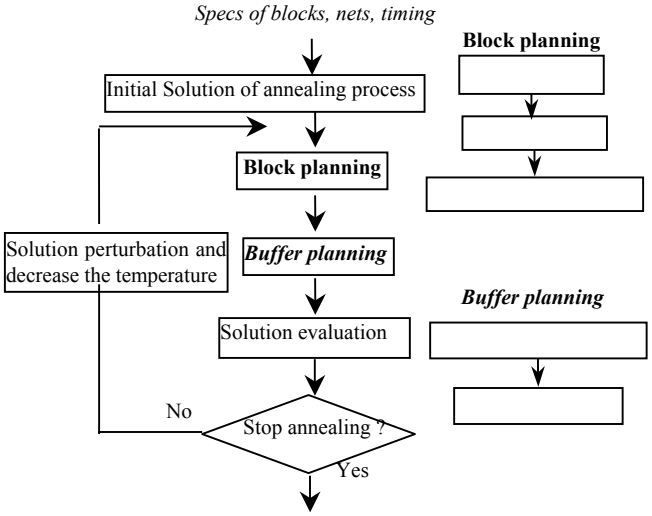


Fig.1. Overall algorithm

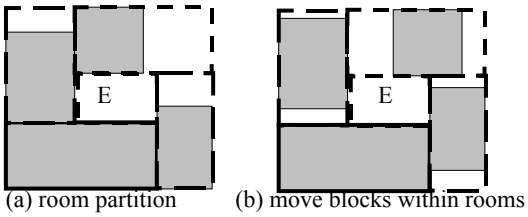


Fig.2. room partition(E is the necessary empty room)

floorplan area, wire length and the timing violations are minimized and the buffers can be inserted in the dead spaces as much as possible and the congestion between routes can be minimized. We assume that the wires are routed over-the-cell. The general flow of our algorithm is given in Fig.1. Our algorithm consists of three key steps: block planning, buffer planning and solution perturbation and evaluation.

The dead spaces besides the blocks will affect the insertion of buffers. Traditional room-based floorplanner will pack the blocks at the lower-left corner or the center of the blocks' rooms. Thus the dead spaces are distributed without considering the buffer insertion demands. Different from the traditional block planning method, to favor the buffer planning, we firstly pack the blocks by given solutions to obtain their room partitions, then the positions of the blocks are determined by the buffer budget in whole packing area. And the later buffer planning is based on the result of the detail locations of the blocks. The congestion information is estimated taken into account the buffer requirement for all nets. The buffers are allocated based on net flow algorithm considering both buffer position and the congestion estimation.

3. Floorplanning Algorithm

In the room-based representation such as (BSG, SP, CBL, Slicing), the blocks are packed within the range of the rooms. If we can move the blocks in their room, the buffer insertion will be optimized since it can leave much more dead space where the buffers are needed.

3.1 Rooms in the packing

The chip can always be dissected into small rectangles, denoted as room(Fig.2(a)). Given an *n*-block set, it divides the chip into at least *n* rooms and assigns no more than one block to each room. And the topological relations between the rooms depend on the representations of the floorplanning. Most of the rooms are not held entirely by the circuit blocks. Therefore, some dead-spaces may be generated. And in the non-slicing structure, some rooms may contain no circuit blocks but dead space so called necessary empty room (Fig.2).

Definition 1: the necessary empty room is the empty room without circuit block and it can not be removed by merging with the other rooms.

The blocks can be moved within their rooms while the area and the topological relations remain. The dead space in necessary empty room should be fixed and the dead space in the block room can be redistributed by moving the block within its room(Fig2(b)).

3.2 Budget of buffer insertion

Each driver/buffer is modeled as a switch-level *RC* circuit[8] and the Elmore delay formula[7] is used for delay computations.

The notation for the physical parameters of the interconnect and buffer we use in this paper is as follows:

r wire resistance per unit length

c wire capacitance pre unit length

T_b intrinsic buffer delay

C_b buffer input capacitance

R_b buffer output resistance

C_L sink capacitance;

R_d driver resistance;

L_n the length of sink-source net N(two pin net)

The optimal locations of the k buffers for delay minimization of the net as shown in [3] are

$$x_i^* = (i-1)y_L^* + x_L^* \qquad i \in \{1,2,...k\} \tag{1}$$

Where

$$x_L^* = \frac{1}{k+1}(L_n + \frac{k(R_b - R_d)}{r} + \frac{(C_L - C_b)}{c})$$

$$y_L^* = \frac{1}{k+1}(L_n - \frac{(R_b - R_d)}{r} + \frac{(C_L - C_b)}{c})$$

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

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

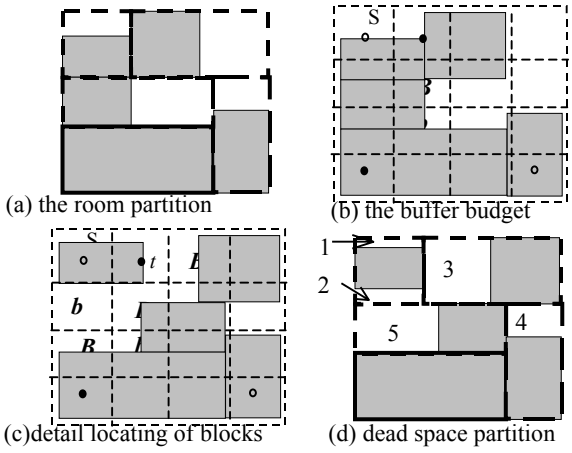
where

$$K_4 = R_bC_b + T_b$$

$$K_5 = (rC_b + cR_b)l + T_b + R_dC_b + R_bC_L - T_{req} - \frac{r}{2c}(C_b - C_L)2 - \frac{c}{2r}(R_b - R_d)^2$$

$$K_6 = \frac{1}{2}rcl^2 + (rC_L + cR_d)l + R_dC_L - T_{req}$$

To budget the buffer insertion, we divide a floorplan into a set of 2-dimensional array of routing tiles. According to formula (1), the optimal position for buffer insertion is only affected by the wire length between source and sink. We assume that the pins are located at the center of the tiles. It is very effective to figure out the possible buffer insertion tiles for different length. As shown in Fig.3, there is a sub-packing with 5 blocks and 2 nets between them. Three possible buffer tiles(B) for net(S,T) and two possible buffer tiles(b) for net(s,t) are shown in Fig.3(b).



(a) the room partition (b) the buffer budget

(c)detail locating of blocks (d) dead space partition

Fig.3 detail location of blocks

Each tile will have more than one buffer inserted. And each buffer may have several possible insertion tiles. Thus we devise a weight for each tile. Assuming that the probability of each possible insertion tile for a buffer in the net are equal, Bi is a buffer of a net and it has Ki possible insertion tiles, which includes tile (x,y). Thus the probability $P(x,y,Bi)$ that the buffer Bi is inserted at tile (x,y) is: $P(x,y,Bi) = \frac{1}{K_i}$

3.3 Detail locating of blocks

To insert the buffers as much as possible, the dead spaces resource should be allocated as needed. Since the blocks can be moved within their rooms, the strategies of how to place the blocks in their rooms will affect the total packing performance. With the budget of the buffer insertion, we introduce a novelty method to place the blocks in their room to favor the buffer insertion. First of all we give the weight for each tile(x,y). Suppose that $BUFF(x,y)$ is the set of all the buffers which can be inserted in tile(x,y)

$$Weight(x,y) = \sum_{B_i \in BUFF(x,y)} P(x,y,Bi)$$

Suppose that the rooms in the packing are $\{R1....Rn\}$. Since the buffer insertion should avoid the circuit blocks, Suppose that $T\_Covered(Ri)$ is the set of the tiles covered by circuit block in room Ri. Some tiles along the circuit block boundary may not be covered entirely. We define the dead space ratio in each tile is

$$DS\_ratio(x,y) = \frac{A_{DS}(x,y)}{A\_Tile}$$

Where $A_{DS}(x,y)$ is the area of dead space in (x,y)

A_Tile is the area for each tile.

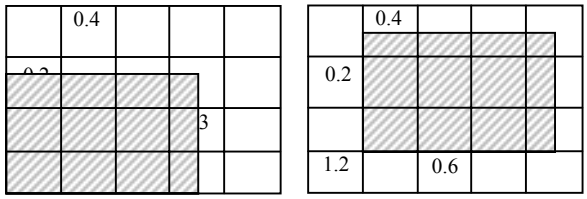
Therefore if tile(x,y) is covered entirely by the circuit blocks, DS_ratio(x,y)=0 and if tile(x,y) is not covered by the circuit blocks at all, DS_ratio(x,y)=1. Thus the object should be to decide the position of the circuit blocks and the buffer budget of the covered tiles is minimized. We define the unused budget in Room Ri as:

$$u\_budget(Ri) = \sum_{(x,y) \in T\_covered(Ri)} (Weight(x,y) * (1 - DS\_ratio(x,y)))$$

In order to optimize the buffer insertion, the unused budget should be minimized, therefore the problem can be described as:

Object: Min $\sum_{i=1}^{n} u\_budget(Ri)$

We have partitioned the packing into rooms, and the budget of buffer insertion are independent in each room. Therefore we can handle the room one by one. As shown in Fig.4, since the tile position is limited in the rooms, we restrict the lower-left corner of the circuit blocks should be located at the tiles. To facilitate the computation, the problem is handled in two directions, one is vertical and the other is horizontal. At first, we can move the



(a) u_b= 1.2*1+0.6*1+0.2*0.6 +0.3*0.5=2.07 (b) u_b = 0.3*1.0+0.4*0.6=0.54

Fig.4. the location of block in one room

block by the size of tile in the vertical direction to decide the vertical position. Then we move the block horizontally to fix the horizontal position. At the same time, the dead spaces in each tile are fixed.

3.4 Effective update of the buffer budget

The detail locating of blocks will change the positions of the blocks slightly. Thus the nets will be influenced at the same time. Since the changes of one block will not affect the buffer insertion greatly, we take a lazy update of the buffer budget. The update process is only taken when the detail locating for a block are fulfilled. Therefore, at the end of the locating process, all the buffer budgets should be updated.

Algorithm 1 *detail locating of blocks*

Budget all the buffer insertions;

Compute the *Weight* and *DS_ratio* for each tile;

For room Ri is from R1 to Rn:

Find the best position of block in room Ri

Update the buffer budget of the nets connecting the block in room Ri

End for

Update the *DS_ratio* for each tile.

End.

4. Buffer Allocation

By taking the congestion information into considered, we use Min-cost-Max-Flow algorithm to assign the buffers to their possible insertion sites.

4.1 Congestion Model

The congestion model employed is essentially a two dimensional rectangular grid based probabilistic map assuming *2-bend* routing for each segment.

Without loss of generality, we consider the source to be located in *tile(* 0, 0) and the sink to be located in *tile(m, n)*. It is easy to see that there are a total of $m + n$ possible *2-bend* routes from source to sink(Fig.5). Without the buffer inserted taken into account, the route number matrix is shown in Table 1.

Taking the buffer sites into consideration, the possible routes should be reduced. Thus we should count only the routes passing through the feasible buffer insertion sites.

Table 1 Routes matrix

Tile(x,y)	#routes
0<x<m, 0<y<n	2
0<x<=m,y=0	(m-x+1)
0<=x<m,y=n	(x+1)
x=0,0<y<=n	(n-y+1)
x=m,0<=y<n	(y+1)
x=0,y=0 or x=m,y=n	(m+n)

Definition 2: **Blocked Tile:** if the tile(x,y) is the possible buffer insertion for a net, but DS_ratio(x,y) = 0 which means tile(x,y) is covered by circuit blocks entirely, the tile(x,y) is called blocked tile.

Since blocked tile can not be routed over, it is necessary to get rid of the routes through tile(x,y) when doing the congestion estimation. Since we are using 2-bend routes, it is very easy to calculate the possible routes by reducing the routes passing through the blocked tile(Fig.6).By scanning all the nets, the congestion estimation for each tile can be figured out.

4.2 Buffer Planning

The objective of the buffer planning is to determine the locations of buffers, and insert as many buffers as possible to maximize the number of nets that meet the timing constraints. To solve the problem, we construct a network graph G (V, E) and then apply a min-cost max flow algorithm to get the solution. Each edge of G represents a possible assignment from a buffer to a tile. G (V, E), V = B ∪ L, where B represents buffers and L represents tiles, E = {(b, l), b ∈ B, l ∈ L, b can be inserted into tile l}. In order to insert as many buffers as possible, we construct an s-t graph based on bipartite graph G to find the max cardinality matchings. To take the congestion into consideration, the capacity and cost for each edge are defined as following:

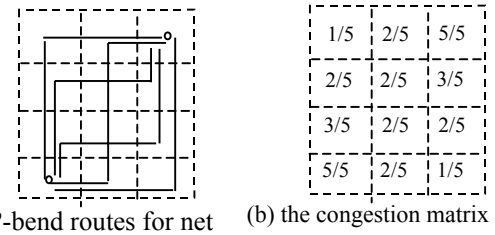
Edge Capacity:

$E(s,b) = 1$; $E(b,l)=1$; $E(l,t)=A_{ds}(l)/A\_buffer$;

Edge cost:

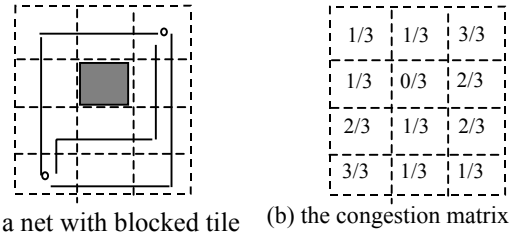
$C(s,b) = 0$; $C(b,l)=0$; $C(l,t)=Congestion(l)$

$A_{ds}(l)$ is the area of dead space in tile l and A_buffer is the area of a single buffer. Hence, $E(l,t)$ records the capacity that how many buffers can be inserted in the dead space within tile l. To punish the high congestion, we use the Congestion(l), which is the congestion estimation of all net in tile l, as the cost for edge (l,t).Fig.7 is an example of the s-t graph.



(a) *2*-bend routes for net (b) the congestion matrix

Fig.5. 2-bend routes and congestion for net



(a) a net with blocked tile (b) the congestion matrix

Fig.6. the routes with block tile

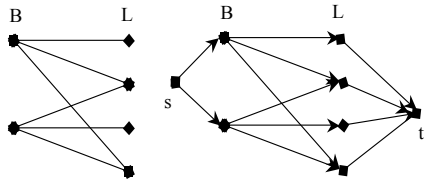


Fig.7. s-t graph of buffer planning: capacities are omitted in the graph

Finding a min-cost maximum flow in a network is a classical problem for which several polynomial-time optimal algorithms are available[9].

5. Two-phase annealing process

In our algorithm, the simulated annealing process is divided into two phases: timing optimization phase and buffer insertion phase. In the timing optimization phase, we try to search for an optimal floorplanning that the timing constraints can be satisfied as much as possible.

In the beginning of floorplanning process, the buffer planning is less meaningful because the locations of the blocks are still far from their final position. The cost function used in the phase is shown below:

*Cost = Area + p*Wire + q* Tviolations*

Where Area is the area of the floorplan and Wire is the total wirelength(p is the weight), Tviolation is the number of the net whose optimal timing with buffer inserted is larger than the given timing constraint.

If the violations of timing constraint is good enough(for example that the violation propotion of timing constraints are less than 25%) or the timing constraint can not be optimized for a long time during the first phase, we start the phase of buffer insertion and the cost function used in the buffer insertion optimization phase is shown below:

*Cost = Area + p*Wire + q* Tviolations + r*Bnot_inserted+m*Congestion*

Here *Bnot_insert* is the number of the buffers not inserted successfully because of the limitation of the dead spaces. *Congestion* is the average of 5% largest congestion estimations in the packing.

6. Experiments

We have implemented the placement algorithm in C programming language, and all experiments are performed on a SUN SPARC III workstation. Some MCNC benchmarks are used in the experiments. The parameters (Table 2) used in our experiments are based on a 0.18um technology in [10]. We have tested our algorithms on 5 MCNC benchmarks, as summarized in Table 3.

In this paper, we focus on 2-pin net, so we decompose each multi-pin net into a set of source-sink 2-pin net. Since the MCNC benchmarks do not come with any timing information, we generate a floorplan by running the CBL floorplanner[11] randomly. Based on this floorplan, we assign target delays to the two-pin nets as follows: for each net, we first compute its best delay by optimal buffer insertion Topt, and assign its target delay as 1.1Topt. Notice that the sizes of the blocks are enlarged for demonstration of the effect of buffer planning.

Table 2 PARAMETER LIST

	Description	Value
R	Wire resistance per unit length(Ωμm)	0.075
C	Wire capacitance per unit length(fF/μm)	0.118
Tb	Intrinsic buffer delay(ps)	36.4
Cs/Cb	Sink/buffer capacitance(fF)	23.4
Rb/Rd	Driver/buffer output resistance(Ω)	180

Table 3 MCNC BENCHMARK

circuit	blocks	nets	2-pin net
apte	9	97	172
xerox	10	203	455
hp	11	83	226
Ami33	33	123	363
Ami49	49	408	545

In the following, the test results are record by : 1)#Meet: the number of nets for which the delay constraint is met with successful buffer insertion; 2) #Inserted B / #B: the ratio of the total number of buffers inserted successfully and the total number of the buffer needed to meet timing constraints, 3)Congestion: the average of 5% largest congestion estimation in the packing.

Based on the fixed placement, the detail distribution of circuit blocks in their rooms can favor the buffer insertion. In Table 4, we give the test result between different strategies to place the blocks within their rooms. In Method LL, all the blocks are located at the lower left corner of the rooms. In Method CE, all the blocks are located at the center of the rooms. The result of method DL is the result of detail locating of blocks. The results show that the detail locating with buffer budget can insert more buffers than the other two methods.

Table 4 The results of detail distribution

Method	Ami33			Ami49		
	#Inserted B	#B	#meet	#Inserted B	#B	#meet
LL	93	245	210	224	532	341
CE	80	253	204	193	524	337
DL	138	266	246	236	486	359

In Table 5, we report the experimental results of two floorplanners: a traditional floorplanner F1 based on simulated annealing without considering timing issue with the buffer insertion, and a timing-driven floorplanner F2 on 2-phase simulated annealing with buffer planning. For the result of F1, we also perform the buffer insertion as in F2 at the end to compare the results of F2:

Comparing F1 and F2 in Table 3, the differences between F1 and F2 on area and wirelength are very small but the timing driven floorplanning algorithm with buffer planning (F2) can achieve much better timing performance than the plain

floorplanning algorithm(F1). We can see some improvement in the results because our algorithm can give a more feasible floorplan structure for buffer insertion. Fig.8 gives a packing result of ami33 with 120 buffers inserted listed in Table 5.

Averagely, the number of the nets which meet their constraints with buffer successful insertion increase 89.8% in F2 than F1. The algorithm of F2 can reduce the total wirelength, constraints violations significantly. The experimental results show that the algorithm F2 can reduce the timing violations efficiently without much expense in area and wirelength.

7. Conclusion

In this paper, the buffer allocation is handled as an integral part in the floorplanning process. By dynamically distribute the blocks in their room according to the buffer insertion budget, we can favor the later buffer planning greatly. Taking the 2-bend routs as the basic model, the congestion information for whole chip scan be estimated taken the buffer insertion into considered. And the buffer allocation in this paper is handled as a net flow problem. Experimental results show that our floorplanner can reduce the timing violation efficiently without much penalty in area and wirelength. Since our algorithm is based on the room-based floorplanning representation, all the room-based representations(such as BSG, SP, CBL, slicing) are fit for this algorithm.

Reference

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

[2] J. Cong, T.Kong, and D. Z. Pan, "Buffer block planning for interconnectdriven floorplanning," in *Proc. Int. Conf. Computer-Aided Design*, Nov.1999, pp. 358–363.

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

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

[5]C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion,"in *Proc. Design Automation Conf.*, June 1997, pp. 588–593.

[6] C.W.Sham, F.Y.Young "Routibility driven floorplanner with buffer block planning", *In ISPD'2002*

[7]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, Jan. 1948.

[8]J. Cong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc.ASP Design Automation Conf.*, Jan. 1999, pp. 97–100.

[9] T. H. Cormen, C. E. Leiserson, R.L. Rivest, Introduction to algorithm, MIT Press.

[10] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*. San Jose, CA: SIA, 1997.

[11]Hong Xianlong, Huang Gang et al. "Corner Block List: An Effective and Efficient Topological Representation of Non-slicing Floorplan" *ICCAD*'2000.pp.8-12.

[12] Yuchun Ma, Xianlong Hong, Sheqin Dong, Song Chen etc "An Integrated Floorplanning with an Efficient Buffer Planning Algorithm" ISPD2003 (in press)

[13] Song Chen, Xianlong Hong, Sheqin Dong, Yuchun Ma, etc "A buffer planningalgorithm based on dead space redistribution", Proceeding of 8th IEEE/ACM Asia &South Pacific Design Automation Conference (ASP-DAC2003).

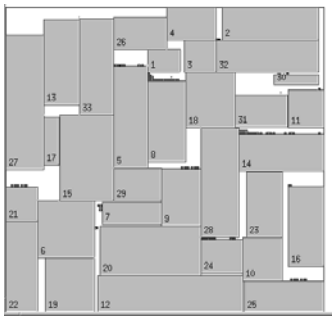


Fig.8. packing of ami33 with buffer inserted

Table 5 Comparison of floorplanning algorithm and the integrated floorplanning algorithm with buffer planning

	[1]Area(mm^2)		[2]Wire(mm)		#Inserted B/#B		#Meet		Congestion		Time(s)	
Test	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2	F1	F2
Xerox_t	20.89	20.89	3389	3358	179/383	340/551	193	374	2.07	2.04	21	86
Ami33_t	1.276	1.299	2619	2596	83/254	120/206	101	255	3.76	3.25	32	305
Ami49_t	38.9	40.1	4676	4683	124/336	111/273	195	359	8.15	7.96	58	643
Apte_t	47.5	47.5	1288	1236	21/121	84/157	83	123	2.52	2.35	6.06	49
Hp_t	9.12	9.53	1884	2028	34/335	98/177	78	163	1.129	1.154	5.3	45
Average	+1.88%		+0.57%		--	--	+89.8%		-4.3%		+725%	

[1] before enlargement [2]for 2-pin nets after enlargement: Note that the wires shorter than the critical length[2,5] should not insert buffers.