A Buffer Planning Algorithm Based on Dead Space Redistribution*

Song Chen[1], Xianlong Hong[1], Sheqin Dong[1], Yuchun Ma[1], Yici Cai[1],
Chung-Kuan Cheng[2], Jun Gu[3]
[1]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[2]*Department of Computer Science and Engineering, University of California,San Diego USA*
[3]*Department of Computer Science, Science & Technology University of HongKong*

Abstract This paper studies the buffer planning problem for interconnect-centric floorplanning for nanometer technologies. The dead-spaces are the spaces within a placement that are not held by any circuit block. In this paper, we proposed a buffer planning algorithm based on dead space redistribution to make good use of dead-spaces for buffer insertion. Associated with circuit blocks under topological representations, the dead space can be redistributed by freely moving some circuit blocks within their rooms in the placement. The total area and the topology of the placement keep unchanged while doing the dead space redistribution. The number of nets satisfying the delay constraint can be increased by redistributing the dead space all over the placement, which has been demonstrated by the experimental results. The increment of the number of nets that satisfy delay constraints is 9% on an average.

I. INTRODUCTION

As the VLSI circuits are scaled into nanometer dimensions and operate in gigahertz frequencies, interconnect design and optimization has become critical. To ensure the timing closure of design, interconnects must be considered as early as in the design flow. A comprehensive survey of interconnect optimization techniques can be found in [4].

Buffer insertion is an effective technique to reduce the interconnect delay. While the Elmore delay of a long wire grows quadratically in terms of the length of the wire, buffer insertion properly results in a linear delay increase due to the length of the wire. The number of buffers needed to achieve timing closure continues to increase with decreasing feature size. Buffers must be planned early in the design because they will take up silicon resources. Recently, many approaches of buffer planning have been proposed.

J. Cong et al[2] introduces the concept of *feasible region*, which is used to generate buffer bocks. Sarkar et al[5] adds the notion of independence to feasible region and tries to improve the routing congestion. Tang and Wong[8] proposes an optimal algorithm assuming only one buffer for each net. F.F. Dragan[9][10] allocates buffers to existent buffer blocks by the multi-commodity flow-based approach. Alpert [11] makes use of tile graph and dynamic programming to perform buffer block planning. They assume that buffers be allowed to be inserted inside macro blocks. Sham[13] proposes a routability driven floorplanner, which can estimate buffer usage and buffer resource for the congestion constraint. F. Rafiq[14] provide an integrated floorplanner with buffer/channel insertion for bus-based microprocessor designs. Although many approaches inserted the buffers into dead space, none of them optimize the distribution of the dead space to improve the number of nets that meet the target delay.

This paper proposes a dead-space redistribution-based buffer planning algorithm to make good use of dead space for buffer insertion. Associated with blocks under topological representations, the dead space can be redistributed by freely moving some blocks within their rooms in the placement, while the total area and the topology of the placement keep unchanged. We compute *independent feasible region* (IFR)[5] for buffer insertion under delay constraint. Each buffer can be inserted into the intersection area between its IFR and the dead-spaces. Therefore, redistributing the dead space in the placement can increase the number of the buffers inserted. In other words, the number of nets satisfying the delay constraints will be increased, which is demonstrated by the experimental results. The increment of the number of nets which satisfy delay constraints is 9% on an average.

The rest of the paper is organized as follows. Section II gives the problem definition, introduces the independent feasible region and gives a brief review of the Corner Block List representation. The redistribution of the dead-spaces is discussed in section III. Section IV proposes the buffer planning and optimization algorithm. Section V and Section VI give the experimental results and conclusion, respectively.

II. PRELIMILARY

A. Problem Definition

In this paper, we concentrate on the buffer planning problem: Given an initial placement/floorplan and timing constraints for each net, we want to determine the number, locations of buffers for each net to satisfy timing closure. The buffers are considered to be inserted into the channel regions or the dead-spaces between circuit blocks. In the following sections, channel regions are also regarded as dead-spaces. Additionally, the dead-spaces are redistributed to maximize the number of nets that meet the target delay.

B. Independent Feasible Region

The concept of *independent feasible region* (IFR) is introduced for buffer insertion [5]. The ***IFR*** for a buffer *b* is the maximum region where *b* can be located such that by inserting buffer *b* into any location in that region, the net delay constraint can be satisfied, assuming that the other buffers of that net are also located within their respective independent feasible regions.

The 2-dimensional feasible region is essentially the union of the one-dimensional ***IFR***s of all possible monotonic Manhattan routes from source to sink, which are convex octilinear polygons, bounded by two parallel lines and the bounding box from source to sink. For example Fig.6 shows a feasible region for a net. The detail of the computation of the ***IFR*** can be found in [5].

*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

C. Corner Block List

Corner Block List (CBL) is introduced in [1]. CBL is a topological representation. Corner Block List represents floorplan by a triple list of (S, L, T), where S stands for block assignment, L and T stand for orthogonal line segments. It dissects the chip into rectangular rooms and assigns one and only one block to each room. Zhou [12] extended the Corner Block List by adding empty rooms into Corner Block List and assigning a dummy block to each empty room. As shown in Fig.2, a dummy block ***0*** is assigned to an empty room. In the following sections, the Corner Block List means the extended CBL except special declaration.

CBL can be constructed from a placement by deleting corner block recursively. The Corner Block is the block assigned to the upper-right corner room of the floorplan. In the floorplan, the segments lying on the left and bottom boundary of the corner block compose a T-junction who defined the orientation of the corner block. The T-junction has alternative orientations: T rotated counterclockwise by 90° (‘+’) and by 180° (‘-’) representing horizontal and vertical orientation, respectively. Fig.2 shows an example.

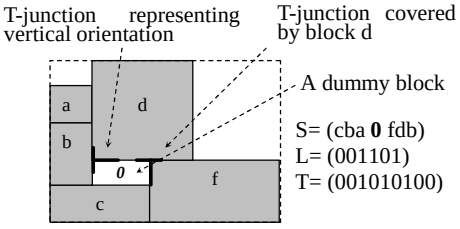


Fig.2. The orientation of block *d* and T-junction covered by *d*

The insertion process of corner block based on given (S, L, T) can construct the corresponding floorplan. A placement and its Corner Block List are shown in Fig.2. The detail of the Corner Block List can be found in [1].

III. REDISTRIBUTION OF THE DEAD-SPACES

The dead-spaces are defined as the spaces within a placement that are not held by any circuit block. The chip can always be dissected into small rectangles, denoted as room, and there is at most one block in each room. All the rooms are not held entirely by the circuit blocks and there may be some empty room assigned no circuit block. Therefore, some dead-spaces may be generated. According to the generation of a dead-space, the dead-spaces in a placement can be classified into the following two types:

Definition 1 If a dead-space is generated because of some empty room, the dead-space is called a **Detached Dead-Space (DDS)**.

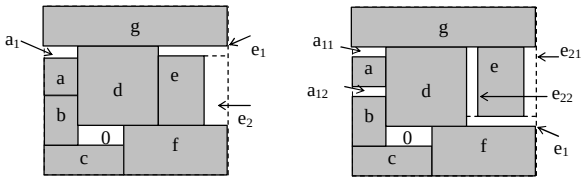
A Detached Dead-Space cannot be associated with any circuit block. For example, the empty room ***0*** shown in Fig.4.(a) is a Detached Dead-Space, and it cannot be associated with any circuit block around it.

Definition 2 A dead-space is called an **Attached Dead-Space (ADS)** if the dead-space is generated because that a room is not entirely held by the circuit block**.**

An Attached Dead-Space can be associated with the circuit block in the room in which the ADS is generated. As shown in Fig.4, the dead-spaces a_1, e_1 and e_2 are Attached Dead-Spaces. The dead space *a1* can be associated with block *a*, and e_1, e_2 can be associated with block *e*.

The following lemma is easily concluded.

Lemma 1 The topology among blocks and the total area of the placement are unchanged when we move a circuit block in its related room.



(a) Dead space type (b) Redistribution

Fig.4. Two type of dead space and Redistribution of the dead space

The Detached Dead-Spaces are moveless, while the Attached Dead-Spaces can be redistributed. Because of the existence of the Attached Dead-Spaces, some circuit blocks can be moved freely in one-dimensional region or 2-dimensional region (such as block *e* in Fig.4.(b)) keeping the topology and the total area unchanged. Through the topological representation of the placement, we can respectively associate Attached Dead-Spaces with some circuit blocks. Consequently, the distribution of dead-spaces can be changed by moving the circuit block within the room which is not “filled”. In Fig.4.(a), the Attached Dead Space e_1 is above block *e*, e_2 is right to block *e*. Fig.4.(b) gives a redistribution of the Attached Dead-Spaces in the placement shown in Fig.4.(a). The dead-space a_1 is divided into a_{11} and a_{12} by moving block *a*, block *e* is moved to divide *e2* into e_{21} and e_{22}, and e_1 is below the block *e* in Fig.4.(b).

Therefore, the following theorem is easily concluded.

Theorem 1 The dead space redistribution can be achieved by redistributing the Attached Dead-Spaces in the placement, while the topology and total area of the placement keep unchanged.

In the following, we describe how to find all the dead-spaces in a placement and to associate each Attached Dead-Space with some circuit block or dummy block under the representation of Corner Block List, which has been introduced in section II.

For a placement of *n* blocks, the CBL dissects the chip into *m* (*m*=*n*) rectangular rooms by horizontal and vertical segments, which determine the topology among rooms. Thus, each of the dead-spaces must be generated in certain room. Because each room is assigned with a circuit block or a dummy block, each dead-space must be associated with some circuit block or dummy block. It is obviously that all the Detached Dead-Spaces must be associated with dummy blocks, and the Attached Dead-Spaces are associated with circuit blocks.

When a corner block is inserted during the packing process, the corner block must cover some other blocks. For each block *b*, we check all the blocks covered by *b*, and determine whether there are dead-spaces between *b* and its covered blocks by comparing the coordinates of them. Simultaneously each dead-space is associated with a circuit block or a dummy

block. For example, blocks *a*, *d*, and *e* are covered by block *g* in Fig.4.(a). We compare the coordinates of block *g* with those of *a*, *d* and *e* to find the dead-spaces between them. The dead-spaces a_1, e_1 and e_2 are found, and dead-space a_1 is a Attached Dead-Space associated with block *a*, dead-spaces e_1 and e_2 are the Attached Dead-Spaces associated with block *e*. By checking the blocks covered by block *d*, we can find a Detached Dead-space ***0*** which is associated with a dummy block. The Attached Dead-Space a_1, e_1 and e_2 can be redistributed since they are associated circuit blocks.

IV. BUFFER PLANNING AND OPTIMIZATION

In this section, we describe the buffer planning algorithm based on the dead-space redistribution in detail. Given a placement, we assume that the buffers can only be inserted into dead space. As shown in [1], the CBL representation of the placement can be obtained by deleting the corner block recursively. And then we compute the dead-spaces in the placement and associate each dead-space with a circuit block or a dummy block using the method described in section III.

A. Buffer Planning

The objective of the buffer planning is to determine the number and locations of buffers, and insert as many buffers as possible to maximize the number of nets meeting the timing constraints.

At first, the candidate tile (shown in Fig.6) set for each buffer is calculated. Secondly a bipartite graph is constructed to represent all the possible assignment from buffers to tiles. Finally the assignment of buffers to tiles is achieved by finding max cardinality matchings in a bipartite graph. Algorithm 1 shows an outline of the algorithm.

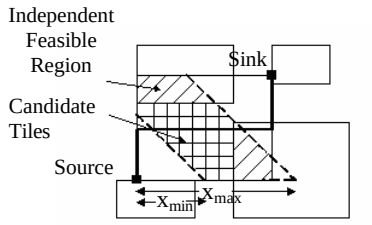


Fig.6. Independent Feasible region and Candidate tiles for a buffer

In step 1, each dead-space is divided into small tiles where the buffers can be located. For each buffer *b*, we compute all the possible tiles that *b* can be placed in step 3. Consequently, the set of all possible buffer assignments is computed, from which a bipartite graph G can be constructed. Each edge of G represents a possible assignment from a buffer to a tile. The bipartite graph G can be defined as follow:

Algorithm 1 Buffer Planning
1.Build the tile data structure for all the dead-spaces.
2.Compute IFR for each buffer.
3.Compute the set of candidate tiles for each buffer
4.Construct a bipartite graph G (V, E), $V = V_1 \cup V_2$, where V_1 represents buffers and V_2 represents tiles, $E = \{(v_1, v_2), v_1 \in V_1, v_2 \in V_2, v_1$ can be inserted into $v_2\}$.
5.Construct an s-t graph from G.
6.Find the max flow from s to t and determine the location of each buffer.

G = (V, E), $V = V_1 \cup V_2$, where V_1 represents buffers and V_2 represents tiles, $E = \{(v_1, v_2), v_1 \in V_1, v_2 \in V_2, v_1$ can be inserted into $v_2\}$.

In step 5, 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. We direct all edges from V_1 to V_2, add a source *s* and a directed edge from *s* to each element of V_1, and add a sink *t* and a directed edge from each element of V_2 to *t*. Let each edge have a capacity 1. The max cardinality matching can be computed by finding the max flow from *s* to *t*.

B. Optimization

The solution of the above Buffer Planning algorithm is optimized by redistributing the dead-spaces all over the placement. The objective is to maximize the number of nets that meet the delay constraints. Algorithm 2 outlines the Optimization algorithm.

Algorithm 2 Optimization
1. Compute all the dead-spaces in the placement and associate each of the dead-spaces with some circuit block.
2. Perform the buffer planning algorithm to compute the number of nets that meet the target delay, denoted as N_{old}.
3. Generate new distribution of the dead-spaces and update related information.
4. Perform the Buffer Planning algorithm to compute the number of nets that satisfy the delay constraints, denoted as N_{new}.
5. If $N_{old} < N_{new}$, the new dead space distribution is accepted, $N_{old} = N_{new}$. Otherwise, restore the changes.
6. Repeat step 3 to step 5 for given times.

The new distribution of the dead space in the placement can be generated by the following two methods.

One method is to randomly select two dead-spaces, and move the selected dead-spaces to the other side of the corresponding circuit blocks. And then we update the length, the buffer number of the nets that have pins in the moved circuit block and independent feasible region of each buffer. In step 5, if the new dead-space distribution is not accepted, we move the selected two dead-spaces to their original locations, and restore the changes in the length and buffer number of the nets and feasible region of buffers.

The other method is to divide a dead-space into two parts, and the associated circuit block will be located between the two parts. In step 5, if the new dead-space distribution is not accepted, we merge the two parts to the original dead-space, and restore the changes in the nets and associated buffers.

It is obviously that the topology of the placement and total area of the chip keep unchanged after the dead-space redistribution. We perform the previous buffer planning algorithm to compute the number of nets that meet the target delay for each new distribution of the dead space.

We cannot ensure that all the nets can satisfy the delay constraints because of the limited dead-spaces. The number of nets that cannot satisfy the target delay through the above Optimization algorithm can be further reduced by a channel-expanded approach as in [2] or [5]. In this paper, we concentrate on the improvement on the number of nets meet the delay constraints by redistributing the dead-spaces, and it is unnecessary to repeat the former work.

V. EXPERIMENTAL RESULTS

The Buffer Planning and Optimization algorithm have been implemented using C language on a SUN Ultra-SPARC

III machine. In this section, we present some details of our experimental results obtained. The values for parameters are based on a 0.18μm technology in the NTRS'97 roadmap[15].

In this paper, we concentrate on solving the problem of buffer planning for two-pin (single source, single sink) nets, and all the multiple-pin nets are decomposed into two-pin nets. Because of the lack of information on signal direction in the benchmark files, we choose a pin to be the source and all the others to be sinks, and then decompose a multiple terminal net into a set of two-pin nets. We ignore all power and ground interconnects. The initial placements of the MCNC benchmark circuits were obtained from [1].

We assign target delays of the two-pin nets, since the MCNC benchmarks include no any timing information. All two-pin nets whose lengths are smaller than the critical length l_{min}[6] are ignored, because buffer insertion cannot reduce their delay. And then we compute the optimal delay *Topt* under optimal buffer insertion [6] for each net and then randomly assign a constraint delay between 1.05 and 1.20 times *Topt* to the net as in [2,5]. Since we generate placements and timing constraints on our own, a direct comparison between our method and that in [2,5] cannot be fair. We provide the results of our algorithm for 5 MCNC benchmark circuits[7]. The details of these circuits are shown in Table 1.

In Table 2, we provide some experimental results from our buffer planning and optimization algorithm, which includes the number of nets which meet the delay constraint, the total number of inserted buffers, the improvement of the optimization algorithm, and the CPU time.

TABLE 1
MCNC BENCHMARKS STATISTICS

Circuit	Blocks	Nets	Two-pin Nets
Apte	9	97	172
Xerox	10	203	455
Hp	11	83	226
Ami33	33	123	363
Ami49	49	408	545

The column of "Buffer Planning" shows the experimental results of running Buffer Planning algorithm under the initial dead space distribution, and the column of "Optimization" is the experimental results of optimization algorithm. The sub-column of "met", "#B", and "time" respectively show the number of nets that satisfy the timing closure, the number of buffers inserted, and the CPU time consumed. The improved number of nets that satisfy the timing constraint and the ratio are respectively given in the column of "N_{imp}" and "R_{imp}".

The results in Table 2 show that Optimization algorithm is able to increase the number of nets that satisfy the delay constraints, while the total area and topology of the placement are unchanged. In circuit Xerox, for example, the number of nets that satisfy the delay constraints is 275 in the initial dead space distribution, and the number increase to 315 after optimization. The nets which satisfy the delay constraints increase 12.4%. For the five circuits, the increment of the number of the nets that satisfy delay constraints is 9% on an average. The experimental results show that our Optimization algorithm is very efficient. Because of the iteration of the dead space redistribution, the run-time of our algorithm is higher than those in [2] and [5].

VI. Conclusion

In this paper, we proposed a buffer planning algorithm based on dead space redistribution to make good use of dead-spaces for buffer insertion. The dead space redistribution can be achieved by redistributing the Attached Dead-Spaces in the placement, while the topology and total area of the placement keep unchanged. Experimental results show that our approach is efficient.

As a basic buffer planning algorithm embedded in the optimization procedure, Our Buffer Planning algorithm can be easily extended to handle the additional constraints, such as congestion and noise. To get an approximate optimal solution, it is necessary to apply an advanced search strategy such as simulated annealing. But because of time consumption, a greedy algorithm is used for optimization in our paper. Though the experimental results show that the greedy strategy is efficient, it is required to develop a faster buffer planning algorithm for applying a better search strategy, and the detour route is not considered in this paper. We will work on it in the future.

TABLE 2
THE RESULTS OF THE BUFFER PLANNING AND OPTIMIZATION ALGORITHM

Circuit	Buffer Planning			Optimization			N_{imp}	R_{imp}
	met	#B	Time (s)	met	#B	Time(s)		
Apte	89	83	0.16	100	104	28.6	11	12.4%
Xerox	275	152	0.1	315	182	8.7	40	14.5%
Hp	129	179	0.25	139	182	25.1	10	7.8%
Ami33	235	162	0.08	249	178	7.1	14	5.9%
Ami49	437	236	0.51	457	253	49.1	20	4.6%

References

[1] X.L. Hong, G. Huang, Y.C. Ma, Yici Cai, S.Q. Dong, "Corner Block List: an effective and efficient topological representation of non-slicing floorplan," ICCAD'2000.

[2] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning", IEEE/ACM ICCAD, 1999.

[3] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies," Frontiers in Semiconductor Research: A collection of SRC Working Papers, Semiconductor Research Corporation, http://www.src.org/prg_mgmt/frontier.dgw, 1997

[4] J. Cong, L. He, C-K. Koh, P.H. Madden, "Performance optimization of VLSI interconnect layout" *Integration, the VLSI Journal*, vol.21, Nov. 1996.

[5] P. Sarkar, C. K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," Intl. Symp. Physical Design, 2000.

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

[7] Collaborative Benchmarking Laboratory, North Carolina State University, *http://www.cbl.ncsu.edu/CBL Docs/lys92.html: LayoutSynth'92 Benchmark Information.*

[8] X. Tang and D.F. Wong, "Planning buffer locations by network flows", Intl. Symp. Physical Design, 2000, pp. 180-185.

[9] F. F. Dragan, A. B. Kahng, I. Mandoiu, S. Muddu, "Provably good global buffering using an available buffer block plan", IEEE/ACM ICCAD, 2000

[10] F. F. Dragan, A. B. Kahng, et al"Provably good global buffering by multiterminal multicommodity flow approximation", ASP-DAC, 2001.

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

[12] Sh. Zhou, S.Q. Dong et al. "ECBL: an extended Corner Block List with solution space including optimum placement", ISPD 2001.

[13] C. W. Sham, F. Y. Young, " Routability driven floorplanner with buffer block planning", ISPD 2002.

[14] F. Ragiq, M. C. Jeske, H. H. Yang, N. Sherwani, "Integrated floorplanning with buffer/channel insertion for bus-based microprocessor designs", ISPD 2002.

[15] Semiconductor Industry Association, National Technology Roadmap for Semiconductors, 1997.