

An Algorithm for Integrated Pin Assignment and Buffer Planning

Hua Xiang and Xiaoping Tang
IBM T. J. Watson Research Center
and Martin D.F. Wong
University of Illinois at Urbana-Champaign

The *buffer block* methodology has become increasingly popular as more and more buffers are needed in deep-submicron design, and it leads to many challenging problems in physical design. In this paper, we present a polynomial-time exact algorithm for integrated pin assignment and buffer planning for all two-pin nets from one macro block (source block) to all other blocks of a given buffer block plan as well as minimizing the total cost $\alpha \cdot W + \beta \cdot R$ for any positive α and β where W is the total wire length and R is the number of buffers. By applying this algorithm iteratively (each time pick one block as the source block), it provides a polynomial-time algorithm for pin assignment and buffer planning for nets among multiple macro blocks. Experimental results demonstrate its efficiency and effectiveness.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids—*Placement and routing*; J.6 [**Computer Applications**]: Computer-Aided Engineering - Computer-Aided Design

General Terms: Algorithms, Design, Performance, Theory

Additional Key Words and Phrases: Buffer insertion, Pin assignment, Min-cost maximum flow

1. INTRODUCTION

As chip size grows larger and minimum feature size is reduced, the capacitance and resistance of wires increase dramatically. This makes interconnects play a critical role in achieving high performance and reducing circuit complexity in deep-submicron design. Many techniques have been proposed to reduce interconnect delay. One effective way is buffer insertion[3][9] and it is heavily used in chip design. It is estimated that the amount of inserted buffers for on-chip interconnect may be up to 800,000 in 50nm technology[4]. At the same time, the introduction of so

This work was partially supported by the National Science Foundation under grant CCR-0306244.

Author's address:

H. Xiang, IBM T.J. Watson Research Center, 1101 Kitchawan road, Rte 134, P.O. Box 218, Yorktown Heights, NY 10598; email: huaxiang@us.ibm.com

X. Tang, IBM T.J. Watson Research Center, 1101 Kitchawan road, Rte 134, P.O. Box 218, Yorktown Heights, NY 10598; email: xtang@us.ibm.com.

M.D.F. Wong, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801; email: mdfwong@uiuc.edu

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20XX ACM 1529-3785/20XX/0700-0001 \$5.00

many buffers raises many challenging problems in physical design. In a hierarchical approach, buffers are clustered together as buffer blocks to facilitate floorplanning and routing. Cong et al.[5], Tang and Wong[12] and Sarkar et al.[11] proposed algorithms for buffer block planning to minimize the chip area and the number of buffer blocks. In two recent works, Dragan et al.[7][8] gave algorithms for global buffered routing problem with fixed pins. Their work is based on multicommodity flow which is an NP-hard problem. In this paper, we address the problem of simultaneous Pin assignment and Buffer planning (PB) for a given buffer block plan. Our algorithm uses min-cost flow computation which is solvable in polynomial time.

Informally, the problem can be described as follows: given a placement of macro blocks and buffer blocks, assign pins and plan buffer insertions for the given set of nets subject to the required lower and upper bounds of connection intervals (i.e., the range of allowable distance between two buffers or two pins or a pin and a buffer) as well as minimizing the total cost $\alpha \cdot W + \beta \cdot R$ where W is the total wire length and R is the number of buffers. Figure 1 shows an example. The placement includes 3 macro blocks, and 3 buffer blocks. Buffer blocks may have different capacities, i.e., the number of buffers in a buffer block can be different. r_1 has a capacity 1 while the capacities of r_2 and r_3 are 2. A net set includes 2 nets between b_1 and b_2 and 1 net between b_2 and b_3 . The range of allowable distance between two buffers or a buffer and a pin is bounded by Manhattan distance 2 and 4. Also if the distance of two pins is longer than 5, buffers have to be inserted, i.e., the lower and upper bounds for the allowable distance between two pins are 0 and 5. The purpose is to assign pins and plan buffers for the 3 nets while minimizing the number of buffers and the total wire length.

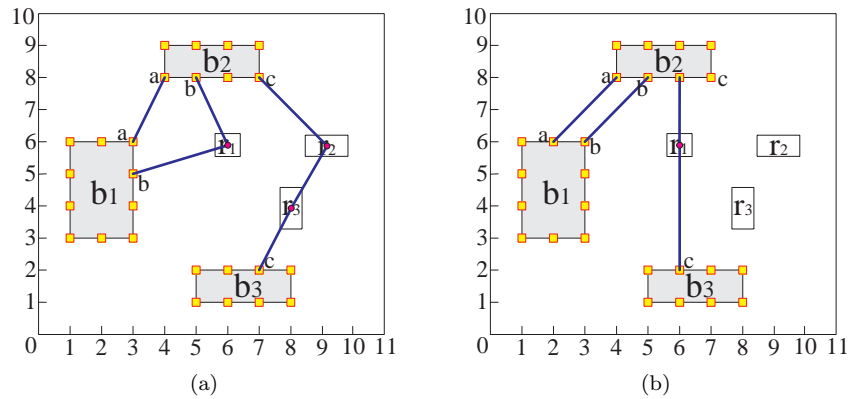


Fig. 1. (a) Three nets use 3 buffers and the total wire length is 20. (b) An optimal solution with 1 buffer and wire length 14.

The goal of pin assignment is to find the exact locations of pins on macro blocks. Buffer planning is to decide buffer usages along net connections to maintain required delay constraints. The two tasks are closely related. Pin assignment alone may neglect many important factors since interconnect is hard to predict, and a global

view of net connections is always helpful. Figure 1(a) illustrates a solution by a two-step approach (i.e., pin assignment following buffer planning). The pin assignments are decided according to the shortest Manhattan distance. Totally 3 buffers are used and the wire length is 20. Figure 1(b) shows an optimal solution with only one buffer and the wire length is 14.

In this paper, we present a polynomial-time exact algorithm for simultaneous pin assignment and buffer planning for all two-pin nets from one macro block (source block) to all other blocks for a given buffer block plan such that each net satisfies the lower and upper bounds on connection intervals as well as minimizing the total cost $\alpha \cdot W + \beta \cdot R$ for any positive constants α and β where W is the total wire length and R is the number of buffers. By applying this algorithm iteratively (each time pick one block as the source block), it provides a polynomial-time algorithm for pin assignment and buffer planning for nets among multiple macro blocks. Experimental results demonstrate its efficiency and effectiveness.

The rest of the paper is organized as follows. Section 2 defines the PBO (Pin assignment and Buffer planning for One source block) problem which simultaneously assigns pins and plans buffers for nets between one macro block and all other blocks. In section 3, we present a network flow formulation to solve the PBO problem. Then we extend PBO problem to PB(Pin assignment and Buffer Planning) problem which considers all nets among multiple macro blocks in section 4. In section 5, we also provide a node clustering method to speed up the computation. Finally, we show the experimental results in section 6 and conclude the paper in section 7.

2. PIN-ASSIGNMENT AND BUFFER PLANNING FOR ONE SOURCE BLOCK (PBO)

Given a placement of $m + 1$ macro blocks $B = \{b_s, b_1, \dots, b_m\}$ with n buffer blocks $R = \{r_1, \dots, r_n\}$. (For convenience, we call b_s source block and other blocks sink blocks.) Each buffer block r_i ($i = 1, \dots, n$) is associated with a positive integer c_i denoting the capacity of r_i , i.e., buffer block r_i can hold c_i buffers.

Let $N = N_1 \cup N_2 \cup \dots \cup N_m$ where N_i ($i = 1, \dots, m$) is the set of nets between block b_s and b_i ; $P = P_s \cup P_1 \cup \dots \cup P_m$ where P_i ($i = s, 1, \dots, m$) is the set of available pin locations of block b_i .

The distance of two points u and v on a planar region is denoted as d_{uv} . Let *buffer interval* be the allowable distance range between two buffers or a pin and a buffer; and *pin interval* be the allowable distance range between two pins. For convenience, let *connection interval* refer to buffer interval or pin interval.

Suppose the lower and upper bounds for buffer intervals are \tilde{L} and \tilde{U} respectively; and those for pin intervals are \bar{L} and \bar{U} respectively. A valid path means:

- (1) If the path is $p = (p_s, r'_1, \dots, r'_k, p_t)$ where $p_s, p_t \in P$ and $r'_i \in R$ ($i = 1, \dots, k$), then the distance of each path segment is bounded by \tilde{L} and \tilde{U} , i.e., $\tilde{L} \leq d_{p_s r'_1} \leq \tilde{U}$, $\tilde{L} \leq d_{r'_i r'_{i+1}} \leq \tilde{U}$ ($i = 1, \dots, k - 1$) and $\bar{L} \leq d_{r'_k p_t} \leq \bar{U}$;
- (2) If the path is $p = (p_s, p_t)$ where $p_s, p_t \in P$, then $\bar{L} \leq d_{p_s p_t} \leq \bar{U}$.

The length of a path is the sum of the distances of all path segments.

For any given positive constants α and β , the PBO problem is to find a set of valid paths connecting b_s and all other macro blocks as well as minimizing the total

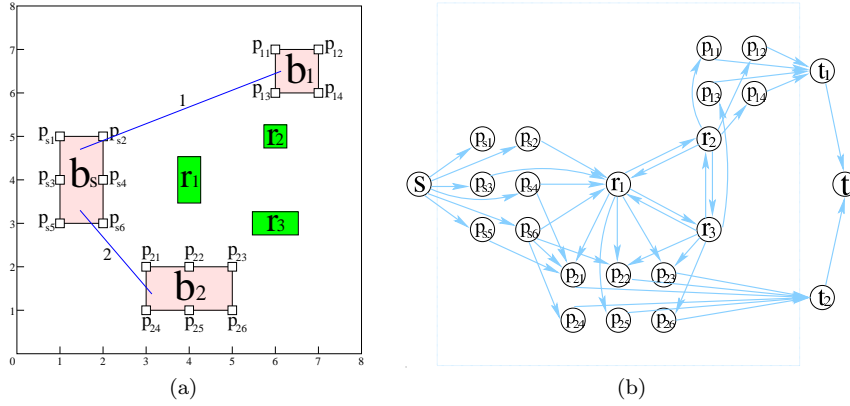


Fig. 2. (a) A PBO problem with 3 macro blocks and 3 buffer blocks. The lower and upper bounds of buffer intervals are 2 and 3 respectively. And the pin intervals are bounded 0 and 3. The tiny squares p_{xy} on the boundaries of macro blocks are available pin locations. The net set includes one net between b_s and b_1 and two between b_s and b_2 . (b) The corresponding flow network graph.

cost $\alpha \cdot W + \beta \cdot R$ where W is the total length and R is the number of buffers. Each path corresponds to a net in N and the two end points of the path are assigned pin locations for this net.

Figure 2(a) gives a simple example. There are 3 macro blocks and 3 buffer blocks. The capacities of the three buffer blocks r_1 , r_2 and r_3 are 2, 1 and 2 respectively. For any two points u and v on the planar region, their coordinates are (u_x, u_y) and (v_x, v_y) respectively. Define distance $d_{uv} = |u_x - v_x| + |u_y - v_y|$. The required lower and upper bounds for buffer intervals are 2 and 3 respectively; and the bounds for pin intervals are 0 and 3 respectively. The purpose is to decide pins and buffer locations for 1 net between b_s and b_1 and 2 nets between b_s and b_2 with a minimum cost $\alpha \cdot W + \beta \cdot R$.

3. THE ALGORITHM

To solve the PBO problem, we first construct a network graph, then apply a min-cost flow algorithm to get the solution.

Given a PBO problem, we construct the network graph $G = (V, E)$ with capacity U and cost C as follows:

- (1) $V = \{s, t, t_1, t_2, \dots, t_m\} \cup R \cup P$, where s is the source node, t is the sink node, and t_i ($i = 1, \dots, m$) is a subsink node.
- (2) $E = E_s \cup E_t \cup E_{\bar{t}} \cup E_b \cup E_r \cup E_{br} \cup E_{rb}$, where

$$E_s = \{(s, p_s) | p_s \in P_s\},$$

$$E_t = \{(t_i, t) | i = 1, 2, \dots, m\},$$

$$E_{\bar{t}} = \bigcup_{i=1}^m \{(p, t_i) | p \in P_i\},$$

$$E_b = \bigcup_{i=1}^m \{(p_s, p) | p_s \in P_s, p \in P_i, \bar{L} \leq d_{p_s p} \leq \bar{U}\},$$

$$E_r = \{(r_i, r_j) | i \neq j, i, j = 1, \dots, n, \tilde{L} \leq d_{r_i r_j} \leq \tilde{U}\},$$

$$E_{br} = \bigcup_{i=1}^n \{(p_s, r_i) | p_s \in P_s, \tilde{L} \leq d_{p_s r_i} \leq \tilde{U}\},$$

$$E_{rb} = \bigcup_{i=1}^n \{(r_i, p) | p \in P_j, j = 1, \dots, m, \bar{L} \leq d_{r_i p} \leq \bar{U}\}$$
- (3) Edge Capacity:

- for edges $e(t_i, t)$, $U(e) = |N_i|, (i = 1, \dots, m)$;
 other edges e , $U(e) = 1$.
- (4) Node Capacity:
 for $r_i \in R$, $U(r_i) = c_i$;
 for $p \in P$, $U(p) = 1$;
 other nodes are incapacitated.
- (5) Edge Cost:
 for $e \in E_s \cup E_t \cup E_{\tilde{t}}$, $C(e) = 0$;
 other edges $e(u, v)$, $C(e) = \alpha \cdot d_{uv}$.
- (6) Node Cost:
 for $r \in R$, $C(r) = \beta$;
 other nodes v , $C(v) = 0$.

Figure 2(b) illustrates the constructed network graph for the PBO problem in Figure 2(a). Note that whether an edge (u, v) ($u, v \in P \cup R$) should be added to G depends on whether d_{uv} falls in the range $[\tilde{L}, \tilde{U}]$ (or $[\bar{L}, \bar{U}]$) or not. For example, the distance between p_{13} and r_2 is 1, which is less than the lower buffer interval bound $\tilde{L} = 2$. Thus there is no edge between p_{13} and r_2 in the constructed flow network. Similarly, the distance between p_{s2} and p_{13} is 5 which is larger than the upper pin interval bound $\bar{U} = 3$, thus the edge (p_{s2}, p_{13}) is not included.

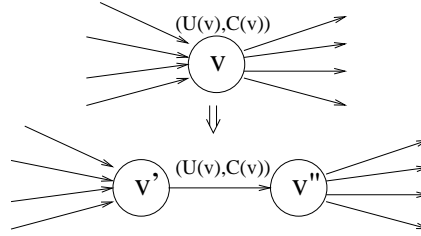


Fig. 3. Node splitting for capacitated nodes. The new edge has capacity $U(v)$ and cost $C(v)$.

In the constructed flow network, every node in $P \cup R$ has a cost and a capacity. However, classical network flow problem only assigns cost and capacity to edges. This can be solved by splitting the capacitated node v into two nodes v' and v'' . A new edge (v', v'') is added with a capacity $U(v)$ and a cost $C(v)$. Then change the original edges (u, v) and (v, w) into edges (u, v') and (v'', w) respectively (refer to Figure 3).

Any flow in G can be mapped to a pin assignment and buffer planning solution for a subset of the given nets. The used capacities of R nodes in the flow are the number of buffers needed in the solution. Figure 4(a) illustrates a flow f , $|f| = 3$. And Figure 4(b) is a solution of pin assignment and buffer planning derived from the above flow. If a flow f exists and $|f| = |N|$, then we can find a feasible solution of pin assignment and buffer planning for all of the nets in N . On the other hand, given a pin assignment and buffer planning solution for n nets, a flow f ($|f| = n$) can always be found on the constructed flow network. Since the total capacities of edges going into sink node t are $\sum_{i=1}^m U(t_i, t) = \sum_{i=1}^m |N_i| = |N|$, the maximum

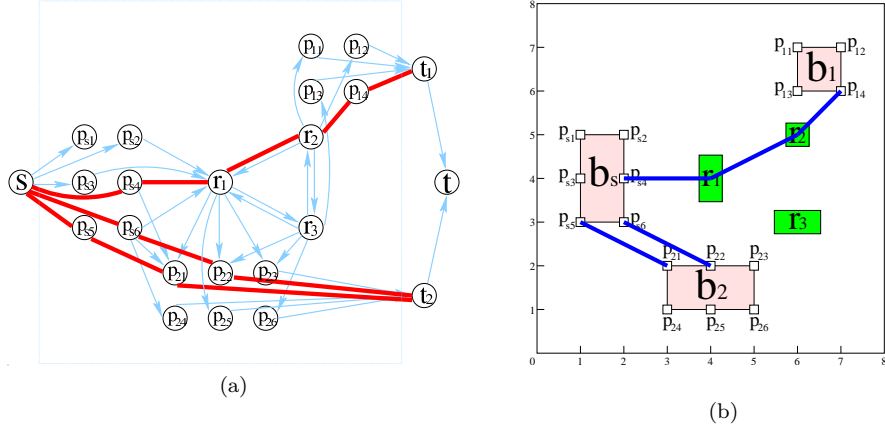


Fig. 4. (a) a flow f in the network in Figure 2(b), $|f| = 3$. (b) The corresponding solution of pin assignment and buffer planning to the PBO problem of Figure 2(a).

flow f_{max} in G , $|f_{max}| \leq |N|$. Thus if a flow $|f| < |N|$, then there is no feasible solution to the original PBO problem, which requires considering all of the nets between b_s and all other macro blocks. Furthermore, the cost of the flow is also the cost of pin assignment and buffer planning solution. Therefore min-cost maximum flow assigns pins and plans buffers for as many nets as possible with minimum total cost.

The following theorem shows that the PBO problem can be exactly solved by min-cost flow computation on G .

THEOREM 1. *A min-cost flow f , $|f| = |N|$, in G corresponds to a pin assignment and buffer planning solution to PBO problem for all nets in N with minimum total cost $\alpha \cdot W + \beta \cdot R$ for any given α and β where W is the total wire length and R is the number of buffers. If the size of the max-flow, $|f_{max}| < |N|$, then there is no feasible solution to the PBO problem. A min-cost maximum flow assigns pins and plans buffers for the maximum number of nets with minimum total cost.*

The algorithm PBO-Flow can be summarized as the following:

Algorithm PBO-Flow($B, R, N, P, C, \alpha, \beta$)

1. Construct the network graph $G(V, E)$
2. Assign capacities U and costs C
3. Apply min-cost maximum flow algorithm on G
4. Derive the pin assignment and buffer planning solution

Finding a min-cost maximum flow in a network is a classical problem for which several polynomial-time optimal algorithms are available[2][6]. Deriving a solution of PBO from a flow in G can be done in $O(E)$ time. Thus, if we adopt the double scaling algorithm in[1], we get the following time complexity for the PBO problem.

THEOREM 2. *The PBO-Flow algorithm optimally solves the PBO problem in $O(VE \log \log U_{max} \log(VC_{max}))$ time for $G = (V, E)$, U_{max} is the maximum value of U , and C_{max} is the maximum value of C .*

Note that the complexity of our PBO-Flow algorithm is mainly dependent on the size of the constructed network graph $G(V, E)$. According to the graph construction rules, $|V| = 2 + m + 2(|P| + |R|)$ (consider node splitting), and the upper bound of edges is $m + |P| + |P_s| \cdot |P| + |P| \cdot |R| + |R|^2 + |P| + |R|$ (in fact, the number of edges is much smaller due to distance constraint) which is bounded by $(|P| + |R|)^2$.

In applications, we may put more effort on reducing the number of buffers. In this case, we can set a large weight to buffer nodes, i.e., a large β . For example, let β larger than the upper distance bound \tilde{U} . If we set β large enough, we tend to get a solution with the minimum number of buffers.

Further in some circuits, some locations on a block may not be allowed for pin assignment. In this case, these kinds of locations will not appear in the pin set P . Obviously, our network-flow based algorithm will not assign a pin to these kinds of locations.

4. PIN ASSIGNMENT AND BUFFER PLANNING (PB)

In the above section, we discuss how to solve PBO problem using min-cost maximum flow computation. PBO problem only consider net connections between one source block and other blocks. In reality, we need to deal with net connections among all of the macro blocks, called PB(Pin assignment and Buffer Planning) problem. The definition of PBO problem can be easily extended to PB problem as the following: Given:

- (1) a placement of m macro blocks $B = \{b_1, b_2, \dots, b_m\}$ and n buffer blocks $R = \{r_1, r_2, \dots, r_n\}$; buffer block r_i has a capacity c_i .
- (2) a set of available pin locations $P = P_1 \cup P_2 \cup \dots \cup P_m$ where P_i ($i = 1, \dots, m$) is a set of available pin locations of macro block b_i .
- (3) a set of nets $\bar{N} = \bar{N}_1 \cup \bar{N}_2 \cup \dots \cup \bar{N}_m$ where \bar{N}_i ($i = 1, \dots, m$) is the set of nets between block b_i and all other blocks.
- (4) two non-negative numbers \tilde{L} and \tilde{U} denoting the lower and upper bound of buffer intervals.
- (5) two non-negative numbers \bar{L} and \bar{U} denoting the lower and upper bound of pin intervals.

Goal:

For any given positive α and β , find a set of valid paths corresponding to the net set \bar{N} as well as minimizing the total cost $\alpha \cdot W + \beta \cdot R$ where W is the total wire length and R is the number of buffers.

PBO-Flow algorithm solves pin assignment and buffer planning for nets between one block and all other blocks. Naturally, if we treat each macro block as the source block and apply PBO-Flow algorithm repeatedly, we can get a solution for all nets among multiple macro blocks. This is the basic idea of PB-Flow algorithm.

Algorithm PB-Flow($B, R, \bar{N}, P, C, \alpha, \beta$)

1. Let b_1 be the source block;
2. Apply PBO-Flow($B, R, \bar{N}_1, P, C, \alpha, \beta$)
3. For $i = 2$ to m
4. Let b_i be the source block;
5. Adjust the network graph $G(V, E)$
6. Apply PBO-Flow($B, R, \bar{N}_i, P, C, \alpha, \beta$)
7. Derive the pin assignment and buffer planning solution

Note that when different blocks are selected as the source block, their constructed flow networks are slightly different, i.e., the edges incident from/to the pin nodes of the two blocks are different. So in PB-Flow algorithm, at each iteration, we need to do some adjustment to transform the existing network graph to the one corresponding to the next source block(line 6).

Now suppose two different macro blocks b_i and b_{i+1} are selected as the source block sequentially. Let G_i be the constructed flow network when b_i is chosen as the source block. The following steps change G_i to b_{i+1} 's corresponding flow network G_{i+1} :

- (1) delete edges connecting the pin nodes of P_i and the pin nodes of other blocks;
- (2) add edges connecting the pin nodes of P_{i+1} and the pin nodes of other blocks as well as maintaining distance constraints.
- (3) reverse the direction of edges (p, r) where $p \in P_i$ and $r \in R$;
- (4) reverse the direction of edges (r, p) where $r \in R$ and $p \in P_{i+1}$;
- (5) reverse the direction of edges (s, p) where $p \in P_i$;
- (6) reverse the direction of edges (p, t_{i+1}) where $p \in P_{i+1}$;
- (7) let t_{i+1} be the source node s and the original source node become a subsink node t_i . Thus remove the edge (t_{i+1}, t) and add one new edge (t_i, t) .

As illustrated in Figure 5, (a) is a PB problem. (b) shows the constructed flow network when b_1 is the source block. When the source block is switched to b_2 , t_2 becomes the source node and the original one becomes a subsink node t_1 . But the edges connecting two buffer nodes remain unchanged.

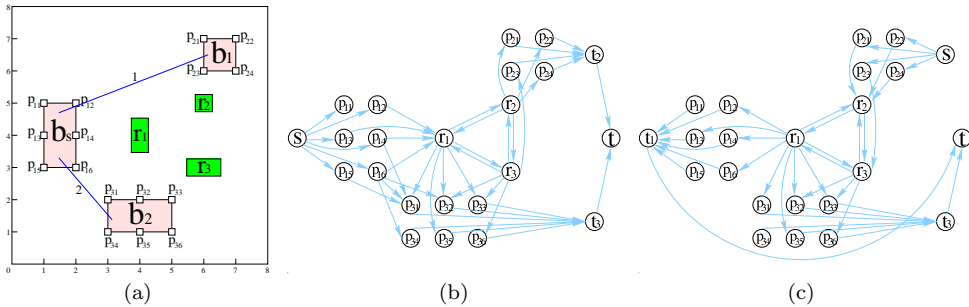


Fig. 5. (a) A PB problem with 3 macro blocks and 3 buffer blocks. (b) The corresponding flow network when b_1 is the source block. (c) The corresponding flow network when b_2 is the source block.

For a newly added edge (u, v) , the cost is $\alpha \cdot d_{uv}$ and the capacity is 1. For other edges (u', v') , the cost is unchanged and the capacity is 1 if no flow flows through the corresponding edge in the previous iteration. Otherwise, the capacity is 0. As to the capacities of nodes, similarly, if a node v has c capacities left after pushing flow in G_i , then the capacity of v is c in G_{i+1} .

G_i and G_{i+1} have the same node set. Also it is easy to show that the number of changed edges is bounded by $(|P_i| + |P_{i+1}|) \cdot (|P| + |R|)$. When taking the distance constraint into consideration, this bound can be further greatly reduced. Thus the adjustment to the graph can be efficiently accomplished.

A two pin net connecting b_i and b_{i+1} belongs to both net sets \bar{N}_i and \bar{N}_{i+1} . Suppose we can get a feasible solution to the PB problem with PB-Flow algorithm. After applying PBO-Flow on G_i , the nets belong to $\bar{N}_i \cap \bar{N}_{i+1}$ should have already been found. Thus we only need to consider $\bar{N}_{i+1} - \bar{N}_i - \dots - \bar{N}_1$ while pushing flows on G_{i+1} .

On the other hand, as we notice that a net belonging to $\bar{N}_i \cap \bar{N}_{i+1}$ should correspond to a path $(p_u, r'_1, \dots, r'_k, p_v)$ ($p_u \in P_i, p_v \in P_{i+1}, r'_i \in R, i = 1, \dots, k$) in G_i . Then in the modified flow network G_{i+1} , the path $(p_v, r'_k, \dots, r'_1, p_u)$ must exist since the edge connections among buffer blocks are not changed. Thus we can remove all paths connected to nodes in P_{i+1} by increasing the capacities along all these paths, and applying PBO-Flow with nets \bar{N}_{i+1} . The optimality of PBO-Flow guarantees that the cost will not increase.

Further even if the algorithm doesn't return a feasible solution, (e.g, no feasible solution exists), the second way still outperforms the first one since the optimality of PBO-Flow assures that the new solution won't become worse: either connecting more nets or reducing the cost.

5. IMPROVEMENT WITH NODE CLUSTERING

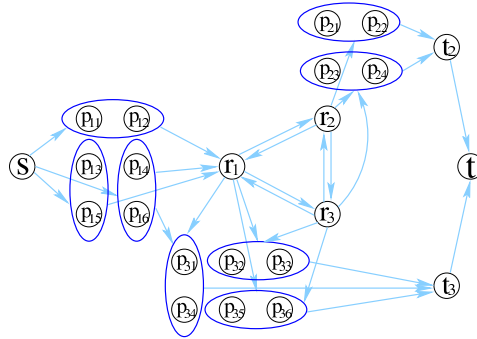


Fig. 6. The corresponding flow network of the PB problem in Figure 5(b) using node clustering method.

When we handle a big circuit which may include a huge amount of nets, the corresponding flow network might be quite large. In order to facilitate the process

of huge PB problems, we propose the following node clustering method to speed up the computation.

For any $p \in P$, if p is inside a macro block, it must be connected to some pin outside the block. Without loss of generality, we may assume that a pin is only on the boundary of a macro block. Then by grouping neighbor pin nodes together, we can greatly reduce the number of nodes, consequently reducing the number of edges. Once several nodes are grouped together, we can use the average coordinate as the location of the new “super-node”. And the capacity of the super-node is the number of nodes it includes. Figure 6 illustrates a constructed flow network for Figure 5(b) when 2 neighbor pins are clustered to one super-node. Actually, we can set different scale rates to blocks according to their sizes or other requirements.

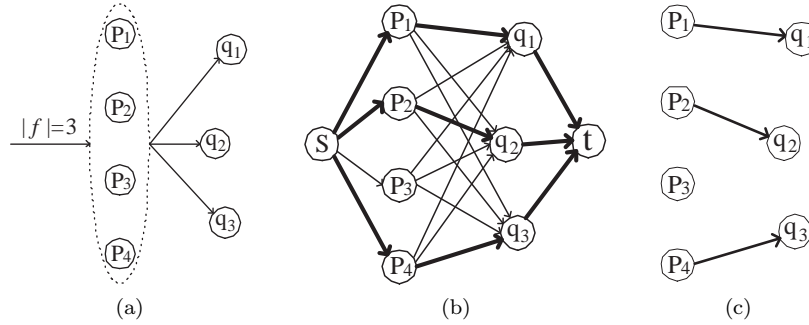


Fig. 7. (a) A flow f , $|f| = 3$ flows through a super-node. (b) The corresponding network and a flow solution. (c) Deriving connections for original pin nodes.

Once we get a solution from the super-node flow network, we need to map the flow to a solution of the original PB problem, i.e, distributing the flows through one super-node to its pin nodes. Of course, we hope the mapping has the minimum connection length. In Figure 7(a), a flow f , $|f| = 3$ flows through a super-node which includes 4 pin nodes. 7(c) shows a feasible mapping. Still it can be solved with min-cost maximum flow as shown in Figure 7(b). Each node p_i ($i = 1, 2, 3, 4$) in the super-node is connected to the destination nodes q_j ($j = 1, 2, 3$) if d_{p_i, q_j} satisfies the distance constraints. The capacity of each edge is 1. The cost of edge (p_i, q_j) is the distance of two nodes d_{p_i, q_j} . All other edges have a cost 0. Min-cost maximum flow guarantees to find an optimal mapping.

6. EXPERIMENTAL RESULTS

Our algorithms were implemented in C++ on PC (733MHz) with 128M memory. We tested PB-Flow on 7 circuits which were generated randomly. For all files, we adopted the node clustering method in Section 5 to reduce runtime.

We compared PB-Flow algorithm with a two-step approach (first assign pins, then plan buffers). We used a classical way[10] to assign pins as follows: by connecting the centers of two macro blocks, we got two crossing points on the boundaries of each block, then assign pins around the crossing points for the nets between these two macro blocks. After pin assignment was done for all nets, we used a net-by-net approach for buffer planning. The net-by-net approach considered only one net

Table I. Average results of PB-Flow for 5 times. All nets are found using PB-Flow algorithm.

File		A33n	X40	H80	S100	F110	M200	T300
Grid		107x106	146x160	235x230	232x227	367x401	587x560	1192x1105
Blocks		33	40	60	91	110	100	99
Buffer Blocks		30	40	120	90	90	120	118
Nets		640	1022	1317	2021	4180	5659	11190
Node Clustering		1 ~ 6	2 ~ 10	1 ~ 16	1 ~ 14	2 ~ 22	2 ~ 34	2 ~ 68
Time (s)	Net-Net	4.25	6.31	12.90	19.68	63.35	88.54	175.33
	PB-Flow	17.26	16.71	45.13	53.36	143.24	212.67	380.95
Found nets	Net-Net	527.6	937.8	1317	1994	4167	5578	11009
	PB-Flow	640	1022	1317	2021	4180	5659	11190
Buffers	Net-Net (per net)	398.6 (0.755)	1732.0 (1.847)	2280.0 (1.731)	6425.0 (3.222)	9961.4 (2.391)	15463.0 (2.772)	30595 (2.779)
	PB-Flow (per net)	323.6 (0.506)	1430 (1.399)	1949.6 (1.480)	5630.4 (2.786)	8639.2 (2.067)	13866.4 (2.450)	27567.2 (2.464)
	Reduced	32.98%	24.26%	14.50%	13.53%	13.55%	11.62%	11.34%
	Wire Length	Net-Net (per net)	926.2 (1.755)	2669.8 (2.847)	3597.0 (2.731)	8419.0 (4.222)	14128.4 (3.391)	21041.0 (3.772)
	PB-Flow (per net)	963.6 (1.506)	2452.0 (2.399)	3266.6 (2.480)	7651.4 (3.786)	12819.2 (3.067)	19525.4 (3.450)	38758.2 (3.464)
	Reduced	14.19%	15.74%	9.19%	10.33%	9.55%	8.54%	8.34%

Table II. Comparison on Node Clustering

File		Time(s)	Buffers	Wire Length
A33n	No Clustering	64.86	307.6	947.6
	2 Node Clustering	18.56	320.0	960.0
	4 Node Clustering	6.56	325.4	965.4
X40	No Clustering	138.44	1401.6	2423.6
	2 Node Clustering	36.37	1423.8	2445.8
	4 Node Clustering	11.33	1436.0	2458.0

each time and found the min-cost path between the two pins of the net for buffer insertion.

For each test circuit, we repeated both approaches 5 times. Table I listed the average results of these 5 times. The block ordering of each run is created randomly, but the solution quality varies very little. For example, for the five runs of the largest test case *T300*, all nets are routed for each run. The total wirelength varies from 38719 to 38808 which are in the range of $\pm 0.1\%$ of the average 38758.2. Also the total number of buffers varies from 27529 to 27618 which are in the range of $\pm 0.2\%$ of the average 27567.8.

Node clustering strategy is applied. The number of nodes to be clustered as a supernode varies since the sizes of macro blocks are different, and the range is listed in “Node Clustering”. Using PB-Flow, we could find a feasible solution of pin assignment and buffer planning for all of the nets with a significant improvement on both the total wire length and the number of buffers. The last two rows show the comparison of the number of buffers and the total wire length. For both methods, we listed the test results and the values per net. The percentage was calculated according to the values per net since the numbers of found nets are different.

Node clustering method speeds up execution, but it may have some effects to solution quality. Therefore, we also compare the running time and solution quality

on the two files $A33n$ and $X40$. The results are listed at Table II. From the table, we can see that by applying node clustering, the running time can be greatly reduced while the increase on wire length or the number of inserted buffers is minor.

7. CONCLUSION

In this paper, we present a polynomial-time algorithm for simultaneous pin assignment and buffer planning for all 2-pin nets between a source macro block and all other blocks such that each net satisfies the lower and upper bound of connection intervals as well as minimizing the total cost $\alpha \cdot W + \beta \cdot R$. By applying this algorithm iteratively (each time pick one block as the source block), it provides a polynomial-time algorithm for pin assignment and buffer planning for nets among multiple macro blocks. Experimental results demonstrate that the algorithm is very efficient and effective.

REFERENCES

- R.K. Ahuja, A.V. Goldberg, J.B. Orlin, and R.E. Tarjan, "Finding minimum-cost flows by double scaling", *Mathematical Programming* 53, pp. 243-266, 1992.
- R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows*, Prentice Hall, 1993.
- H.B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.
- J.Cong, "Challenges and opportunities for design innovations in nanometer technologies", SRC Working Papers, http://www.src.org/prg_mgmt/frontier.dgw, Dec 1997.
- J.Cong, T.Kong, and D.Z. Pan, "Buffer block planning for interconnect driven floorplanning", *Proc. ICCAD*, pp. 358-363, Nov.1999.
- T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, 1992.
- F.F. Dragan, A.B.Kahng, I.I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering using an available buffer block plan", *Proc. ICCAD*, pp. 104-109, 2000.
- F.F. Dragan, A.B.Kahng, I.I. Mandoiu, S. Muddu, and A. Zelikovsky, "Provably good global buffering by multiterminal multicommodity flow approximation", *Proc. ASP-DAC*, pp. 120-125, 2001.
- R.Otten, "Global wires harmful?", *Proc. ISPD*, pp. 104-109, 1998.
- B.Preas and M.Lorenzetti, "Physical Design Automation of VLSI Systems", Benjamin/Cummings, Menlo Park, CA 1988.
- P.Sarkar, V. Sundararaman and C.K. Koh, "Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning", *Proc. ISPD*, April 2000.
- X.Tang and D.F. Wong, "Planning buffer locations by network flows", *Proc. ISPD*, pp. 180-185, April 2000.