Bus-Driven Floorplanning *

Hua Xiang
CS Dept, UIUC
Urbana, IL 61801
huaxiang@uiuc.edu

Xiaoping Tang
Cadence Design Systems
San Jose, CA 95134
xtang@cadence.com

Martin D.F. Wong
ECE Dept, UIUC
Urbana, IL 61801
mdfwong@uiuc.edu

Abstract

In this paper, we present an integrated approach to floorplanning and bus planning, i.e., bus-driven floorplanning (BDF). We are given a set of circuit blocks and the bus specifications (i.e., the net list of blocks for the buses). A feasible BDF solution is a placement of all circuit blocks such that each bus can be realized as a rectangular strip (horizontal or vertical) going through all the blocks connected by the bus. The objective is to determine a feasible BDF solution that minimizes floorplan area and total bus area. Our approach is based upon the sequence-pair floorplan representation. After a careful analysis of the relationship between bus ordering and block ordering in the floorplan represented by a sequence pair, we derive feasibility conditions on sequence pairs that give feasible BDF solutions. Experimental results demonstrate the efficiency and effectiveness of our algorithm.

1. Introduction

As the deep submicron technology advances, chips become more congested even though more metal layers are used for routing. Usually a chip includes several buses. As design increases in complexity, bus routing becomes a heavy task, especially for networking chips or data processors. Since buses have different widths and go through several module blocks, the positions of macro blocks greatly affect bus planning. To ease bus routing and avoid unnecessary iterations in physical design, we need to consider bus planning in early floorplanning stage.

In this paper, we address the problem of bus-driven floorplanning (BDF). We use top two layers for bus planning, and buses go either horizontally or vertically on one layer in floorplanning stage. The simple bus structure is good and efficient at planning stage, and would facilitate bus routing in later stages. Furthermore, more complicated bus structure can always be decomposed into several segments of horizontal/vertical buses.

Informally, the problem can be described as follows. Given a set of rectangular macro blocks and the bus specifications (i.e., the net list of blocks for the buses), find a placement of all circuit blocks such that each bus can be realized as a rectangular strip (horizontal or vertical) going through all the blocks connected by the bus. At the same time, the chip area as well as the total bus area is minimized.

Figure 1 gives an example. Figure 1 (a) and (b) are two floorplans with the same chip size. Two buses u (A, C) and v (B, E, H) are placed in the floorplan of Figure 1 (a). However, neither of the buses can be assigned based on the floorplan in Figure 1 (b) since blocks B, E, H are not aligned and the vertical overlap between blocks A and C is less than the width of bus u.

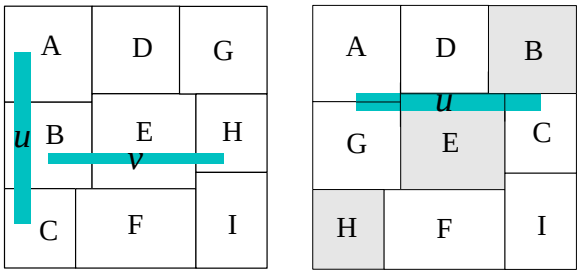

Figure 1: Two floorplans have the same chip size. (a) Two buses u (A, C) and v (B, E, H) are assigned. (b) Neither of the buses can be assigned since B, E, H are not aligned and the vertical overlap between A and C is less than the width of bus u.

In some previous works, researchers have discussed some particular kinds of floorplan constraints related to alignment. However, these kinds of alignment constraints are not suitable for bus-driven floorplanning. The paper [8] handles a kind of alignment in which modules involved in an alignment are required to be aligned by left (right/bottom/upper) side. But this is not necessary in BDF problems. For example, the bottom sides of blocks B, E, H are not aligned, but bus v still fits in the floorplan in Figure 1 (a). Tang et al. [7] proposed another alignment constraint in which several blocks are aligned in a row, abutting with each other. But blocks involved in one bus do not need to be placed adjacent to each other. In Figure 1 (a), A and C are not adjacent while bus u is assigned. The paper [1] discussed predefined coordinate alignment constraint in which some blocks are to be placed along a predefined coordinate within a small region. In BDF, there are no constraints on coordinates. Rafiq et al. [3, 4] proposed bus-based integrated floorplanning. However, the bus defined in their works is composed of bundles of wires connecting only two blocks. Also bus assignment is accomplished by global routing.

Most floorplan algorithms use simulated annealing to search for an optimal solution. The implementation of simulated annealing scheme depends on a floorplan representation where a neighbor solution is generated and examined by perturbing the representation (called 'move'). In this paper, we use sequence pair representation and analyze the relationship between bus ordering and sequence pair representation. Then a fast evaluation algorithm is proposed to transform a sequence pair representation to a floorplan with buses inserted. Finally, we also develop an efficient algorithm to handle soft modules to further improve solution quality. Experimental

*This work was partially supported by the National Science Foundation under the grant CCR-0244236 and by a grant from the Cadence Design System Inc.

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

results on three sets of test files (MCNC benchmarks, industry test files, bus grid test files) demonstrate the effectiveness and efficiency of our approach.

The rest of the paper is organized as follows. Section 2 provides background information on the sequence pair representation. The formal definition of BDF problem is given in section 3. In section 4, we analyze the relationship between bus ordering and sequence pair representation. Then a fast evaluation algorithm is proposed to transform a sequence pair to a BDF solution in section 5. In section 6, a simulated annealing BDF algorithm is presented. Finally, we address how to handle soft blocks to improve solution quality in section 7. Experimental results are given in section 8, and section 9 concludes the paper.

2. Preliminary

A sequence pair is a pair of sequences of n elements representing a list of n blocks. In general, a sequence pair imposes the relationship between any two blocks a and b as follows.

(i) If a is ahead of b in both sequences, a is to the left of b in the floorplan.

(ii) If a is ahead of b in the first sequence while behind b in the second sequence, a is below b in the floorplan.

The original paper which proposed sequence pair [2] presented an algorithm to transform a sequence pair to a floorplan in $\Theta(n^2)$ time. Recently, Tang et al. sped up the evaluation algorithm to $O(n\log n)$ in [5], and later further to $O(n\log\log n)$ in [6].

[5, 6] showed that the coordinates of blocks and the width and height of a floorplan can be obtained by computing longest common subsequence (LCS) in terms of the two sequences. Given a sequence pair (X,Y), the width of a floorplan equals the length of the longest common subsequence of X and Y where weights are blocks' widths. Furthermore, given a block b, let $(X,Y) = (X_1bX_2, Y_1bY_2)$ and $LCS(X,Y)$ be the length of the longest common subsequence of (X,Y). Then the x-coordinate of block b equals $LCS(X_1,Y_1)$ with blocks' widths as weights. Similarly, the height of a floorplan is determined by dealing with the longest common subsequence of (X,Y^R) where Y^R is the reverse of Y and weights are blocks' heights. Furthermore, all the computations of blocks' x/y coordinates can be integrated into a single longest common subsequence computation for a sequence pair.

3. Problem Formulation

Suppose the routing region has multiple layers and buses can be assigned on the top two layers. So the orientation of buses is either horizontal or vertical. The problem of bus-driven floorplanning (BDF) can be defined as follows.

Definition 1. Bus-Driven Floorplanning (BDF)

Given n rectangular macro blocks $B = \{b_i | i = 1,...,n\}$ and m buses $U = \{u_i | i = 1,...,m\}$, each bus u_i has a width t_i and goes through a set of blocks B_i where $B_i \subseteq B$ and $|B_i| = k_i$. Decide the positions of macro blocks and buses such that there is no overlaps between any two blocks or between any two horizontal (vertical) buses, and bus u_i goes through all of its k_i blocks. At the same time, the chip area as well as the total bus area is minimized.

In BDF problems, buses should go through all of their related blocks. So the positions of blocks greatly affect bus assignment. For convenience, let $< g,t,\{b_1,...,b_k\} >$ represent a bus u where $g \in \{H,V\}$ is the orientation, t is the bus width, and b_i $(i = 1,...,k)$ are the blocks the bus goes through. For short, a bus is just represented as $\{b_1,...,b_k\}$. Also let (x_i,y_i) be the lower-left corner of block b_i. And the width and height of block b_i are w_i and h_i respectively. In the following, we give the necessary conditions of feasible horizontal and vertical bus respectively.

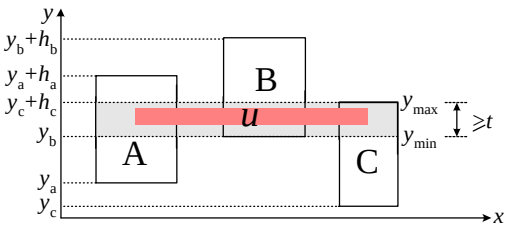

Figure 2: A feasible horizontal bus $u =< H,t,\{A,B,C\} >$. $y_{max} = y_c + h_c$, $y_{min} = y_b$ and $y_{max} - y_{min} \geq t$.

Lemma 1. Feasible Horizontal Bus (H-Bus) *If a horizontal bus $u =< H,t,\{b_1,...,b_k\} >$ is feasible, then $y_{max} - y_{min} \geq t$ where $y_{max} = \min\{y_i + h_i | i = 1,2,...,k\}$ and $y_{min} = \max\{y_i | i = 1,2,...,k\}$.*

Lemma 2. Feasible Vertical Bus (V-Bus) *If a vertical bus $u =< V,t,\{b_1,...,b_k\} >$ is feasible, then $x_{max} - x_{min} \geq t$ where $x_{max} = \min\{x_i + w_i | i = 1,2,...,k\}$ and $x_{min} = \max\{x_i | i = 1,2,...,k\}$.*

Figure 2 illustrates an H-bus $u =< H,t,\{A,B,C\} >$. In order to fit in bus u, the vertical overlap of the three blocks has to be larger than the bus width t.

4. Bus Ordering via Sequence Pair

Sequence pair always entails a packing if no constraints are given. However, when constraints are introduced, there may not exist corresponding packing for some sequence pairs.

In this section, we discuss the relationship between bus ordering and sequence pair representation. First, a necessary condition is derived when only one bus is considered. Then we discuss the relative position of any two horizontal (vertical) buses imposed by a sequence pair. Based on the analysis of two bus ordering, we set up a bus ordering constraint graph and propose an algorithm to remove infeasible buses.

4.1 A Necessary Condition for One Bus

Since blocks cannot overlap in a BDF solution, blocks have at most one-dimension overlap, i.e., if the projections on x-axis of two blocks have overlap, their projections on y-axis cannot overlap; on the other hand, if the projections on y-axis of two blocks have overlap, their projections on x-axis cannot overlap. However, in order to fit in a bus $\{b_1,...,b_k\}$, the projections on x-axis (y-axis) of b_i and b_j $(i,j = 1,...,k; i \neq j)$ must have overlap. In other words, the position relationship of any two related blocks has to be left-right (below-above) . Thus we have the following necessary condition.

Theorem 1. (Block Ordering) *Given a sequence pair (X,Y) and a bus $u = \{b_1,...,b_k\}$, if u is feasible, then the ordering of the*

k blocks should be either the same or reverse in the two sequences X and Y. Furthermore, if the k blocks appear in the same order in both X and Y, the orientation of the bus is horizontal; otherwise the bus is vertical.

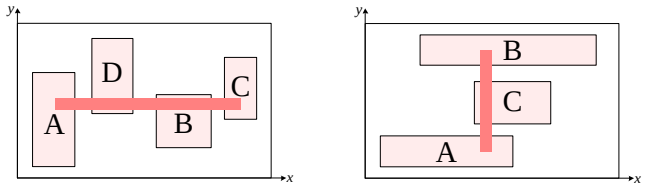

Figure 3: A necessary condition for one bus. (a) The sequence pair must be $(\ldots A \ldots D \ldots B \ldots C \ldots\ ,\ \ldots A \ldots D \ldots B \ldots C \ldots)$ to fit in a horizontal bus $\{A,B,C,D\}$. (b) The sequence pair must be $(\ldots A \ldots C \ldots B \ldots\ ,\ \ldots B \ldots C \ldots A \ldots)$ to fit in a vertical bus $\{A,B,C\}$.

For convenience, this necessary condition is also called *block ordering*. Figure 3 gives two examples. The sequence pair for (a) is $(\ldots A \ldots D \ldots B \ldots C \ldots, \ldots A \ldots D \ldots B \ldots C \ldots)$, and a horizontal bus $\{A,B,C,D\}$ can be assigned. (b) shows another example. The sequence pair is $(\ldots A \ldots C \ldots B \ldots\ ,\ \ldots B \ldots C \ldots A \ldots)$ and the bus is a vertical one $\{A,B,C\}$. Note that Theorem 1 deals with only one bus. When multiple buses are considered, it is likely that some buses cannot be assigned for the floorplan although each bus satisfies the necessary condition.

4.2 Bus Ordering between Two Buses

The relative position of blocks is determined by a sequence pair. Since buses go through blocks, the ordering of buses is also influenced by the sequence pair.

Given a sequence pair (X,Y) and two horizontal buses $u = \{a_1, a_2, \ldots, a_k\}$ and $v = \{b_1, b_2, \ldots, b_l\}$, denote the block set $S_u = \{a_1, a_2, \ldots, a_k\}$, $S_v = \{b_1, b_2, \ldots, b_l\}$, and $S = S_u \cup S_v$. Suppose $|S| = L$ ($L \leq k+l$ since the two buses may go through the same block) and (X,Y) satisfies *block ordering* for the two buses. Also we assume these L blocks appear in the sequence pair as $(\ldots c_1 \ldots c_2 \ldots \ldots c_L \ldots,$ $\ldots d_1 \ldots d_2 \ldots \ldots d_L \ldots)$ where $c_i \in S$ and $d_i \in S$ $(i = 1, \ldots, L)$, and the subsequence pair $(X',Y') = (c_1\ c_2\ \ldots\ c_L\ ,\ d_1\ d_2\ \ldots\ d_L)$. For convenience, let $p[c_i] = i$ $(i = 1, \ldots, L)$ which denotes the position of c_i in X', and $q[d_i] = i$ representing the position of d_i in Y'. From this subsequence pair (X',Y'), we can derive the relative position of the two buses.

Case 1. If $\forall a \in S_u$, $p[a] \geq q[a]$, and $\exists a \in S_u$, $p[a] > q[a]$, then bus u is above bus v.

Suppose $p[a_i] > q[a_i]$, then (X,Y) must be $(\ldots b_j \ldots a_i \ldots\ ,\ \ldots a_i \ldots b_j \ldots)$. b_j is below a_i. Since u goes through a_i while v goes through b_j, bus u is above v.

Figure 4 (Case 1) shows an example. The subsequence pair is $(D\ A\ E\ B\ F\ C,\ A\ D\ B\ E\ C\ F)$. $p[A] = 2$ and $q[A] = 1$; $p[B] = 4$ and $q[B] = 3$; $p[C] = 6$ and $q[C] = 5$. So bus $u = \{A,B,C\}$ is above bus $v = \{D,E,F\}$.

Case 2. If $\forall a \in S_u$, $p[a] \leq q[a]$, and $\exists a \in S_u$, $p[a] < q[a]$, then bus u is below bus v.

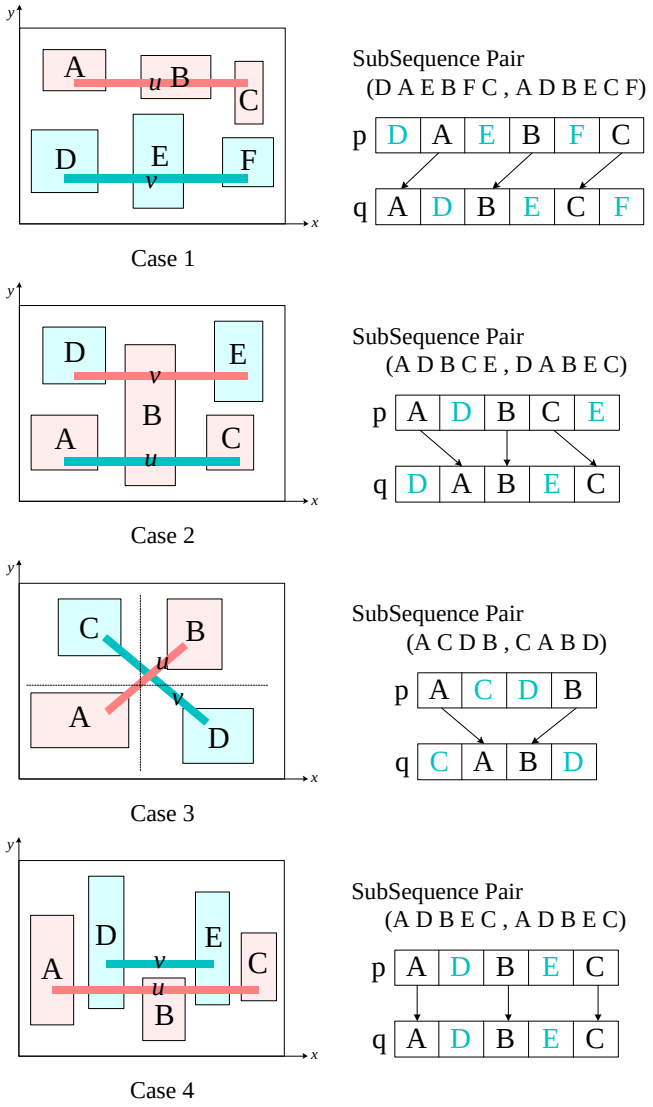

Figure 4: Cases of relative positions of two horizontal buses.

Figure 4 (Case 2) shows an example. Block B is shared by both buses. Bus $u = \{A,B,C\}$ is below bus $v = \{D,B,E\}$.

Case 3. If $\exists a \in S_u$, $p[a] > q[a]$, and $\exists a' \in S_u$, $p[a'] < q[a']$, then the two buses u and v cannot be assigned at the same time.

Suppose $p[a_i] > q[a_i]$ and $p[a_j] < q[a_j]$, then (X,Y) must be $(\ldots b_I \ldots a_i \ldots a_j \ldots b_J \ldots\ ,\ldots a_i \ldots b_I \ldots b_J \ldots a_j \ldots)$. Block b_I is below a_i while b_J is above a_j. The positions of blocks are illustrated in Figure 4 (Case 3).

In the example, the two buses are $u = \{A,B\}$ and $v = \{C,D\}$. Then the subsequence pair is (X',Y') is $(A\ C\ D\ B,\ C\ A\ B\ D)$. For block A, $p[A] = 1$ and $q[A] = 2$ while $p[B] = 4$ and $q[B] = 3$. In this case, the two buses cannot be assigned at the same time.

Case 4. If $\forall a \in S_u$, $p[a] = q[a]$, then the two buses have no firm ordering. Either bus can be above the other.

Figure 4 (Case 4) illustrates an example. In this example, bus $u = \{A,B,C\}$ can be below bus $v = \{D,E\}$. On the other hand, u is

also possible above v. Therefore, the two buses have no bus ordering constraint.

For any two vertical buses, we can get the similar results from (X, Y^R).

4.3 Multiple Bus Ordering

In a BDF solution, it is impossible that the ordering of several buses forms a cycle. For example, bus u is above bus v, bus v is above bus w, while bus w is above bus u. This kind of relationship cannot exist in a feasible solution.

In the above section, we have discussed bus ordering imposed by the given sequence pair. To express the relative positions among buses, we construct bus ordering constraint graphs for horizontal buses and vertical buses respectively. The construction rules for a horizontal bus ordering constraint graph are listed as follows. The graph for vertical buses can be derived similarly.

- Each bus is represented by a node.
- If one bus u is above another bus v (Case 1 or 2) , add one edge (u, v).
- If one block related to bus u is above a block related to bus v, while another block related to u is below a block related to v (Case 3), add two edges (u, v) and (v, u).
- If two buses have no bus ordering constraint (Case 4), no edge is added.

The horizontal bus ordering constraint graph serves in two ways: (i) Given a BDF solution, the block packing must correspond to a sequence pair. Then the horizontal bus relationship imposed by the sequence pair can be represented by an acyclic constraint graph. (ii) If a constraint graph contains a cycle, then at least one bus cannot be assigned. According to the construction rules, there are two kinds of cycles.

1. A cycle includes only two nodes. Then the relative position of the two corresponding buses must comply with Case 3, and at least one bus cannot be assigned. Figure 5 (a) shows an example. Two buses $u = \{A, B\}$ and $v = \{C, D\}$ are crossing, and the subgraph is given in Figure 5 (b).
2. A cycle includes at least three nodes. Figure 5 (c) illustrates an example. There are 3 buses $u = \{A_1, A_2\}$, $v = \{B_1, B_2\}$ and $w = \{C_1, C_2\}$. The sequence pair is $(... A_1 ... B_1 ... B_2 ... C_1 ... C_2 ... A_2 ... , ... B_1 ... A_1 ...C_1 ... B_2 ... A_2 ... C_2 ...)$ From this sequence pair, we can conclude that bus u should be below v, v should be below w, and w should be below u. However, this is impossible in a BDF solution. Therefore, at least one bus has to be discarded.

If a bus ordering constraint graph contains cycles, there must be some buses that can not be assigned. Since our target is to assign as many buses as possible, the problem becomes how to remove minimum number of buses so that the graph is acyclic. However this problem is a NP-Complete problem.

For convenience, if some nodes are removed from the graph $G = (V, E)$, then edges connecting to/from these nodes are also removed, and the result graph is called residual graph. Also all nodes are indexed, and node $u < v$ means the index of u is less than that of v.

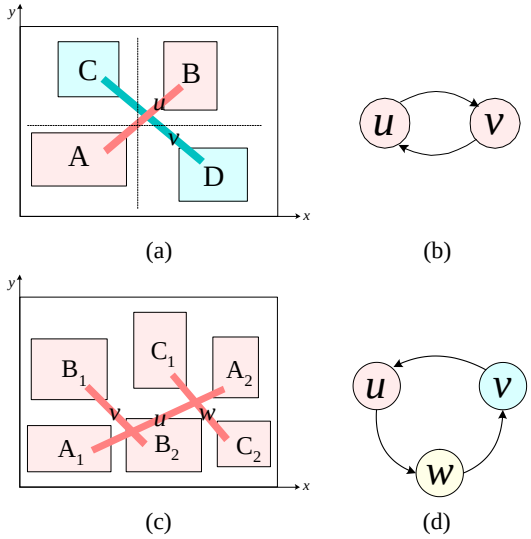

Figure 5: Two kinds of cycles in bus ordering constraint graph.

Definition 2. Node-Deleting Problem (NDP)
Given a sequence pair and a set of buses, a horizontal (vertical) bus ordering constraint graph can be constructed. Remove nodes from the constraint graph so that the residual graph is acyclic. At the same time, the number of deleted nodes is minimized.

Theorem 2. *Node-Deleting Problem (NDP) is NP-hard.*

Proof First NDP is a NP problem since whether a residual graph is acyclic or not can be judged in $O(|V_r|^3)$.

Next we prove that Independent Set Problem (ISP) is polynomial-time reducible to NDP, i.e., ISP $\leq_p$ NDP. Since ISP is NP-Complete, we can conclude that NDP is also NP-Complete.

Let $G = (V, E)$ be an instance of ISP. Suppose $|V| = N$ and $|E| = M$. We form a directed graph $G_d = (V, E_d)$ where $E_d = \{(i, j)|(i, j) \in E\} \cup \{(j, i)|(i, j) \in E\}$, i.e., each edge in G is represented by a pair of edges with different directions in G_d. Figure 6 shows an example. (a) is an instance of independent set problem G. (b) is G_d. Note that G is an undirected graph.

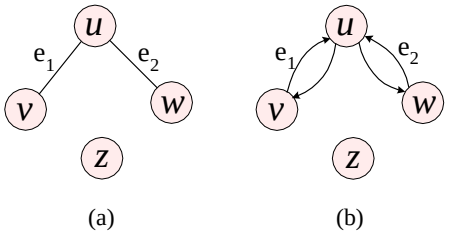

Figure 6: Independent Set Problem and Node Deleting Problem. (a) An instance of independent set problem (ISP). (b) G_d is a horizontal bus ordering constraint graph.

Given a node subset $\bar{V}$, we can get a residual graph $\bar{G_d}$ of G_d by removing nodes in $\{V - \bar{V}\}$. We show that $\bar{V}$ is an independent set of G if and only if the residual graph $\bar{G_d}$ is acyclic. If $\bar{V}$ is an independent set of G, then there are no edges in $\bar{G_d}$. Obviously, $\bar{G_d}$ is acyclic. On the other hand, for any residual graph $\bar{G_d}$ of

G_d, if it contains no cycles, then there are no edges in $\bar{G}_d$ since edges also appear in pairs. Therefore, the nodes in $\bar{G}_d$ also form an independent set of G.

Finally we show that G_d is a bus ordering constraint graph.

For any edge $e_i = (u, v) \in E$ $(u < v)$, let block sequence $x_i = (a_i^u\ b_i^v\ c_i^v\ d_i^u)$, and block sequence $y_i = (b_i^v\ a_i^u\ d_i^u\ c_i^v)$, where a_i^u, b_i^v, c_i^v and d_i^u are macro blocks. Suppose there are L independent nodes w_i $(i = 1, ..., L)$ in G which are not incident on any edge. Let block sequence $x_{M+i} = (a_{M+i}^w\ d_{M+i}^w)$ and block sequence $y_{M+i} = (a_{M+i}^w\ d_{M+i}^w)$ where a_{M+i}^w and d_{M+i}^w are blocks.

We form a sequence pair $(X, Y) = (x_1 ... x_M x_{M+1} ... x_{M+L}\ , y_1 ... y_M\ y_{M+1} ... y_{M+L})$. Blocks in X and Y form the macro block set B. So totally there are $4M + 2L$ blocks. Since there are N nodes in G, the number of buses is also N. For each bus p, the blocks that p goes through are $\{z_i^p | z_i^p \in B, i = 1, ..., (M+L), z = a, b, c, d\}$. Since the ordering of blocks of each bus is always the same in both sequences, all buses are horizontal buses.

For each pair of buses u and v ($u < v$), we get the subsequence pair $(X', Y') = (x'_1 x'_2 ... x'_M, y'_1 y'_2 ... y'_M)$, where x'_i is a subsequence of x_i, and y'_i is a subsequence of y_i. Blocks appearing in X' or Y' are related to either bus u or v. Furthermore, (x'_i, y'_i) $(i = 1...M)$ can be only one of the two cases.

(i) $(x'_i, y'_i) = (x_i, y_i)$;

(ii) $x'_i = y'_i$.

If $\exists J$, $(x'_J, y'_J) = (x_J, y_J)$, then J is unique since there is only one edge between two nodes u and v in G. At the same time, the subsequence pair (X', Y') involves bus crossing (Case 3) for bus u and v. Therefore, the bus constraint graph contains two edges (u, v) and (v, u).

On the other hand, if $\forall i \in \{1, ..., (M+L)\}$, $x'_i = y'_i$, the two buses have no bus ordering constraint and there is no edge between the two nodes u and v in the constraint graph. Also if $x'_i = y'_i$ (x'_i/y'_i can be empty), then there is no edge between u and v in G either. Thus we can conclude that G_d is the horizontal bus ordering constraint graph for the constructed sequence pair. □

Since NDP is NP-Complete, we derive a heuristic method to remove nodes from a graph so that the residual graph is acyclic. The method is based on the following lemma.

Lemma 3. *Given a directed graph, if for each node, its in-degree and out-degree are both non-zero, then each node must be incident on a cycle in the graph.*

The proof is easy and we omit it in the paper. The following is the algorithm.

Algorithm Node_Deleting (V, E)
1. for i = 1 to $|V|$
2. Calculate in-degree and out-degree of nodes in V
3. Find min in-degree min_{in} and min out-degree min_{out}
4. if (min_{in} = 0) or (min_{out} = 0)
5. Remove the corresponding node v from V
6. Remove edges connecting to/from v from E
7. else
8. Find the node v with max degree
9. Insert v into Remove_Set
10. Remove v from V and related edges from E
11. endfor
12. return Remove_Set

For each iteration, the size of V is reduced by one. If we can find a node whose in-degree or out-degree is 0, then this node is treated as a good node. Otherwise, we select the node v with max degree (in-degree + out-degree) and insert it to the Remove_Set, i.e., v should be discarded in order to break cycles. This algorithm guarantees that if the graph is acyclic, Remove_Set is empty. The running time is $O(|V|^2)$.

Figure 7 shows an example. Suppose there are 9 buses and the constraint graph G is Figure 7 (a).We first remove nodes whose in-degree or out-degree is zero. a, d and e are removed from G as Figure 7 (b). Then for the rest of the nodes, they must be incident on a cycle, and we have to delete nodes in order to break cycles. Since c has the maximum degree, c is deleted and make b and g free. The result is illustrated as Figure 7 (c). Finally i is deleted to break the cycle between i and h. Therefore, there are two nodes c and i in Remove_Set. Figure 7 (d) shows the residual acyclic graph after deleting c and i. Furthermore, based on (d), it is easy to find a bus ordering consistent with the below-above relationship imposed by the sequence pair. For instance, the bus ordering (from bottom to top) could be g, d, f, h, a, b, e.

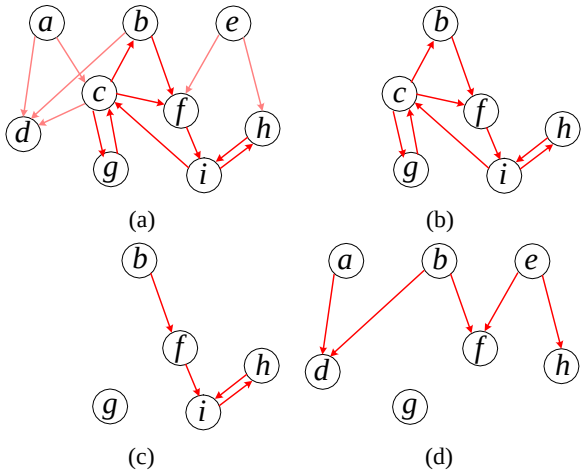

Figure 7: Node Deleting Algorithm. (a) An instance of bus ordering constraint graph G. (b) Nodes whose in-degree or out-degree is zero are removed from G. (c) Node c is deleted from G in order to break cycles. (d) The residual acyclic graph of G after deleting c and i.

5. Evaluation Algorithm

The evaluation algorithm is to transform a sequence pair representation to a BDF solution. However, for some sequence pairs, it is impossible to fit in all of the buses. For example, if a sequence pair violates *block ordering* for a bus, then some buses can not be assigned. Therefore, the target of the evaluation algorithm is to find a floorplan which assigns as many buses as possible. The algorithm is summarized as follows. Suppose there are n blocks and m buses.

Algorithm Evaluation_BDF(*Seq*, *Bus*)
1. Feasible_Bus_Checking_Orientation
2. Bus_Ordering
3. Modified_LCS_Computation

In the following, we explain the above three procedures one by one.

Feasible_Bus_Checking_Orientation

According to Theorem 1, if the blocks of a bus violate *block ordering* in the given sequence pair, the bus cannot be assigned. Therefore, the first step is to identify these buses and remove them from the bus set. For each bus, one scan of the sequence pair is enough to make the judgment. At the same time, if blocks related to a bus appear in the same order in both sequences, the bus is a horizontal bus, otherwise, the bus is a vertical one. This step takes $O(mn)$ time.

Bus_Ordering

Due to the bus ordering imposed by the given sequence pair, some buses can not be assigned at the same time as discussed in section 4. We apply Node_Deleting algorithm to further remove some buses. Meanwhile, since the constraint graph is acyclic, we sort the horizontal buses from bottom to top (from left to right for vertical buses) according to the below-above (left-right) relationship. This bus order will be used in the next step. This step takes $O(m^2)$ time.

Modified_LCS_Computation

The algorithm is based on the engine of computing longest common subsequence (LCS) presented in [6]. LCS computation calculates x coordinates and y coordinates separately. And it always pack blocks from bottom to top (from left to right). In this section, we only discuss the calculation of y coordinates of blocks with the assignment of horizontal buses. The calculation of x coordinates of blocks and vertical buses can be derived similarly.

For any given horizontal bus $< H, t, \{b_1, ..., b_k\} >$, the y coordinates of the k blocks are first calculated with LCS computation. Then these k blocks are aligned so that the bus can be inserted. Suppose the height of b_i is h_i and the y-coordinate of the lower-left corner of b_i is y_i. The basic alignment can be performed as follows.

$y_{max} = \max\{y_i | i = 1, 2, ..., k\}$

$y_i = \max(y_i, y_{max} + t - h_i) \qquad \forall i \in \{i = 1, 2, ..., k\}$.

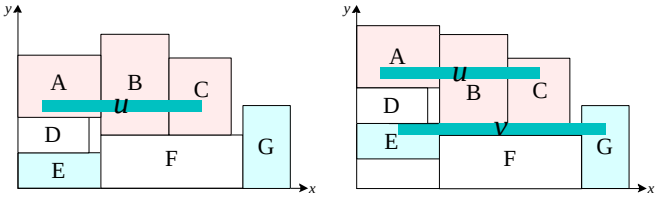

Figure 8: Insert two horizontal buses to the floorplan represented by $(E\ D\ A\ F\ B\ C\ G, A\ D\ E\ B\ C\ F\ G)$. (a) One horizontal bus $\{A, B, C\}$ is assigned. (b) In order to insert another bus $\{B, E, G\}$, blocks A and D have to move up and this makes the bus $\{A, B, C\}$ changed, too.

However, the alignment adjustment for different buses may affect each other. For example, in Figure 8, the given sequence pair is $(E\ D\ A\ F\ B\ C\ G, A\ D\ E\ B\ C\ F\ G)$, and two buses $u = \{A, B, C\}$ and $v = \{B, E, G\}$ are to be inserted. Suppose bus u is first assigned as Figure 8 (a). When we want to place bus v, block E has to move up, which causes blocks A and D to move up too. Furthermore, due to the change of block A, bus u has to be reassigned. Since LCS computation packs blocks from the bottom up, it is important to process buses in the same way. Each time, the lowest bus is selected and assigned so that the bus would not be affected by later processing of other buses. By calling Bus_Ordering procedure, a sorted bus list can be obtained.

At the same time, when multiple horizontal buses are considered, we also need to avoid overlaps between horizontal buses. Therefore, the above basic alignment is not enough.

For example, in Figure 9, two horizontal buses $u = \{C, D\}$ and $v = \{B, E\}$ are to be assigned. First, bus u is assigned with y_u as the y-coordinate of its bottom edge. However, according to the basic alignment calculation, the y-coordinate of bus v's bottom edge is also y_u. Then bus u and v are overlapped as Figure 9 (a). This is not allowed in a BDF solution. We call this situation *Bus_Overlap*.

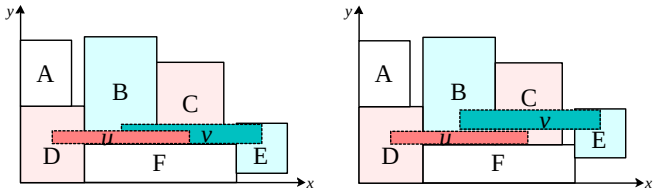

Figure 9: Insert two horizontal buses $u = \{C, D\}$ and $v = \{B, E\}$ to the floorplan. (a) Two buses overlap due to basic alignment adjustment. (b) The assignment of two buses without overlap.

If two buses have bus ordering constraint (only Case 1 and 2 in Section 4.2 are considered since only one bus can be assigned in Case 3), the above situation can not happen. This is because in Case 1 or 2, there must exist a block of one bus which is above a block of the other bus. Then the two buses cannot overlap. On the other hand, if two buses have no bus ordering constraint (Case 4 in Section 4.2), *Bus_Overlap* may happen.

Suppose two horizontal buses $u = \{a_1, a_2, ..., a_k\}$ and $v = \{b_1, b_2, ..., b_l\}$ have no bus ordering constraint. Denote the block set $S_u = \{a_1, a_2, ..., a_k\}$, $S_v = \{b_1, b_2, ..., b_l\}$. Still there are three cases.

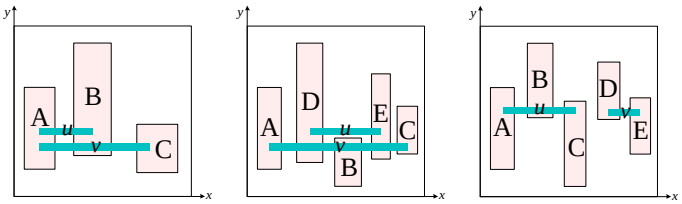

Figure 10: (a) Two buses $u = \{A, B\}$ and $v = \{A, B, C\}$ share blocks A and B. *Bus_Overlap* may happen. (b) The blocks of two buses $u = \{A, B, C\}$ and $v = \{D, E\}$ appear interlaced along x-axis. *Bus_Overlap* may happen. (c) Two buses $u = \{A, B, C\}$ and $v = \{D, E\}$ have no overlaps along x-axis. *Bus_Overlap* is impossible.

Case A $S_u \cap S_v \neq \phi$. Two buses share at least one block. *Bus_Overlap* may happen, like Figure 10 (Case A) which involves two buses $u = \{A, B\}$ and $v = \{A, B, C\}$.

Case B $S_u \cap S_v = \phi$, and the subsequence pair (X', Y') is $(\ldots a_{i_1} \ldots b_j \ldots a_{i_k} \ldots, \ldots a_{i_1} \ldots b_j \ldots a_{i_k} \ldots)$. Figure 10 (Case B) shows an example. In this case, the blocks of two buses appear interlaced along x-axis. It is likely that *Bus_Overlap* happens.

Case C $S_u \cap S_v = \phi$, and in the subsequence pair (X', Y'), all blocks in S_u appear ahead of (behind) blocks in S_v. Figure 10 (Case C) shows an example. In this case, the two buses do not have overlap along x-axis. Therefore, the y-coordinates of the two buses can be decided independently, i.e., *Bus_Overlap* cannot happen.

Based on the above discussion, we can conclude that only Case A and B can lead to *Bus_Overlap*. Therefore, in the basic alignment calculation, we also need to check Case A and B so that no overlaps between two buses are introduced. This kind of checking can be incorporated in Bus_Ordering procedure, and keep the results in a table. If Case A or B is detected, the current bus has to move up until there is no overlap with the buses below.

In summary, after we get the bus list from Bus_Ordering procedure, we apply LCS computation m times (suppose there are m buses in the bus list), In one iteration, after the positions of blocks related to bus u are calculated by LCS, we check if bus u has Case A or B with buses below u and do the move up alignment accordingly. Once the position of a bus is calculated, the bus is not changed any more. Therefore, each iteration fixes one bus. The running time of LCS is $O(n \log\log n)$ [6] where n is the number of blocks. So Modified_LCS_Computation is bounded by $O(mn \log\log n + m^2)$.

6. BDF Algorithm

Most floorplan algorithms based on sequence pair representation use simulated annealing (SA). In this paper, we also use SA to search an optimal or near optimal solution to a BDF problem.

Perturbation (Move)

We use the following operations to generate a neighbor sequence pair in simulated annealing.

Swap is to swap two blocks in either the first sequence or the second sequence. Swap can be done in constant time.

Rotation is to rotate a block (e.g. exchange the width and height of a block). Rotation does not cause any changes to the sequence pair. This operation can be done in constant time.

Cost Function

The target of BDF problem is to minimize the chip area and the total bus area. At the same time, we hope to insert all of the buses. Therefore, we define the cost function as follows.

$$Cost = \alpha \cdot C + \beta \cdot B + \gamma \cdot M$$

where C is the chip area, B is the bus area and M is the number of unassigned buses, α, β and γ are coefficients and defined by users.

By applying Evaluation_BDF Algorithm, the positions of blocks and buses are all calculated. Therefore, it is easy to get the value of C, B and M.

7. Soft Block Adjustment

For floorplan, the shapes of some blocks may not be fixed. For example, for some blocks, their areas are fixed, but their width/height ratio can be changed in some range. This kind of blocks are called *soft block*. The flexibility of soft block shapes can help us improve BDF solution quality further. Our strategy is as follows.

After applying BDF algorithm, we get a BDF solution. Based on this solution, each time, we select a soft block on LCS path which decides the size of the chip [5, 6]. Reduce the width (height) of the block a little bit, and apply Modified_LCS_Computation to get a new solution. This process is executed repeated.

In order to control iterations, simulated annealing is adopted again. The perturbation operation is to choose a soft block on LCS path and change its width or height accordingly. The cost function is the same as that of BDF algorithm. Figure 11 shows an example. Figure 11 (a) shows a BDF solution. Blocks B, D and E are on an LCS path. E is selected and its width is reduced. Figure 11 (b) illustrates the BDF solution after repacking. Both chip area and bus area are reduced.

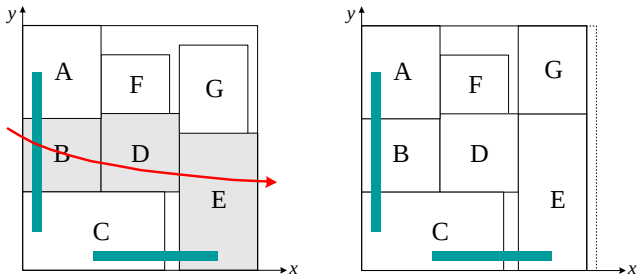

Figure 11: Soft Block Adjustment. (a) A BDF solution. Block E is on an LCS path. (b) The new BDF solution after changing the shape of block E.

8. Experimental Results

Our algorithm was implemented in C++ and tested on workstations (2.4GHz) with 1G memory. The technique of simulated annealing is used to search for an optimal or near optimal BDF solution with a special annealing schedule where a very large number of temperatures are used but only a small number of moves are made within each temperature.

Table 1: Test Set 1

File	Block	Bus	Results		Soft-Adjust	
			time (s)	dead space	time (s)	dead space
apte	9	5	11	4.11%	12 (+1)	0.72%
xerox	10	6	12	3.88%	13 (+1)	0.95%
hp	11	14	28	5.02%	28 (+0)	0.62%
ami33-1	33	8	61	6.02%	62 (+1)	0.94%
ami33-2	33	18	81	6.10%	86 (+5)	1.27%
ami49-1	49	9	98	5.42%	101 (+3)	0.85%
ami49-2	49	12	278	6.09%	281 (+3)	0.84%
ami49-3	49	15	265	7.40%	268 (+3)	1.09%

We tested on three sets of test files. The first set is derived from MCNC benchmarks for block placement. We added different numbers of buses to the benchmarks. Once we got a BDF solution, we also applied the soft block adjustment technique to further improve the solution quality. Furthermore, we also made some post-processing to move buses apart from each other. The test results are

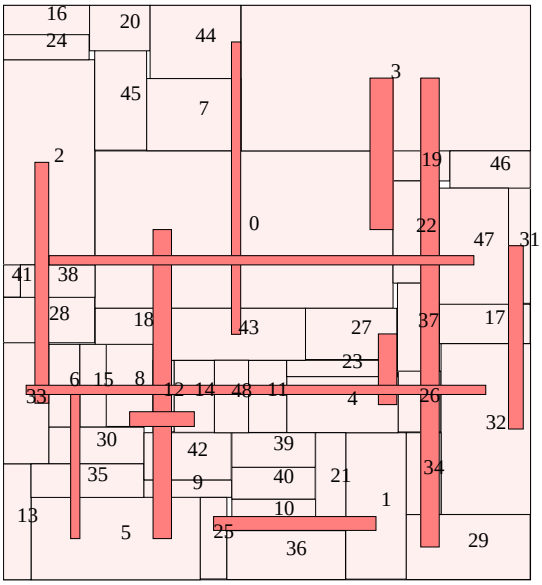

Figure 12: The result packing of ami49-2 after soft block adjustment. Totally, there are 49 blocks and 12 buses. The buses are $\{0,5,9,12,18\}$, $\{1,10,21,25\}$, $\{2,28,33\}$, $\{3,19,22,26,29,34\}$, $\{4,23,27\}$, $\{5,35,30,6\}$, $\{32,31,17\}$, $\{11,14,15,32,33\}$, $\{12,8,14\}$, $\{44,43,7\}$, $\{0,3\}$, $\{2,47\}$.

listed in Table 1. As illustration, Figure 12 displays the final packing result of ami49-2 after soft block adjustment. The second test set (cad1 and cad2) includes two test files which are derived from industry designs. To further test our approach, we created a set of bus grid test files. Each test file includes n^2 $(n = 4, ..., 7)$ blocks and $2n$ buses. All blocks have the same block size (600×700) and each bus goes through n blocks. This kind of problems is quite hard and the position of one bus heavily affects the assignment of other buses. Still our approach can find an optimal solution within a short time. Figure 13 illustrates an optimal solution to the test file grid7. In this solution, not only the chip area is minimized, but also the total bus area is minimized.

Table 2: Test Sets 2 & 3

File	Block	Bus	time (s)	deadspace
cad1	40	13	209	4.40%
cad2	57	16	191	5.16%
grid4	16	8	1	0%
grid5	25	10	23	0%
grid6	36	12	103	0%
grid7	49	14	150	0%

9. Conclusion

In this paper, we consider the bus-driven floorplanning (BDF) problem. We first derive necessary conditions for feasible buses. Then based on the analysis of the relationship between bus ordering and sequence pair representation, we develop an efficient evaluation algorithm which transforms a sequence pair representation to a BDF solution. Simulated annealing is adopted to search for an optimal or near optimal BDF solution. We also propose a simple but efficient way to handle soft blocks. Experimental results demonstrate that our approach is very efficient and effective.

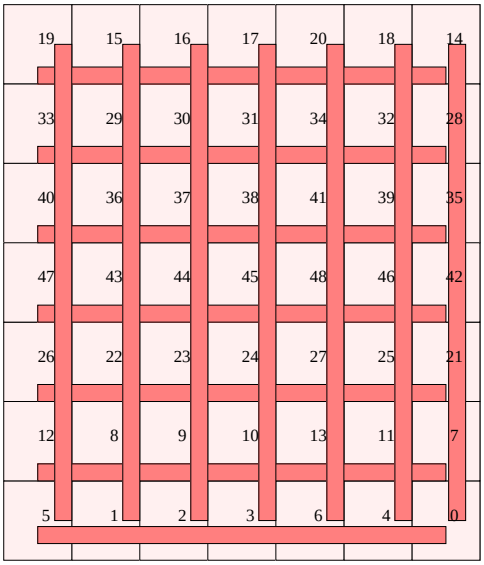

Figure 13: An optimal packing of grid7. There are 49 blocks and 14 buses. The buses are $\{0,1,2,3,4,5,6\}$, $\{7,8,9,10,11,12,13\}$, $\{14,15,16,17,18,19,20\}$, $\{21,22,23,24,25,26,27\}$, $\{28,29,30,31,32,33,34\}$, $\{35,36,37,38,39,40,41\}$, $\{42,43,44,45,46,47,48\}$, $\{0,7,14,21,28,35,42\}$, $\{1,8,15,22,29,36,43\}$, $\{2,9,16,23,30,37,44\}$, $\{3,10,17,24,31,38,45\}$, $\{4,11,18,25,32,39,46\}$, $\{5,12,19,26,33,40,47\}$, $\{6,13,20,27,34,41,48\}$.

10. References

[1] R. Liu, X. Hong, S. Dong Y. Cai and J. Gu. "VLSI/PCB placement with predefined coordinate alignment constraint based on sequence pair", Proceedings. 4th International Conference on ASIC, pp. 167-170, 2001.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. "VLSI module placement based on rectangle-packing by the sequence-pair", IEEE Transaction on CAD, vol. 15:12, pp. 1518-1524, 1996.

[3] F. Rafiq, M. Chrzanowska-Jeske, H. H. Yang, N. Sherwani. "Bus-based integrated floorplanning" IEEE International Symposium on Circuits and Systems, pp. 875 -878, 2002.

[4] F. Rafiq, M. Chrzanowska-Jeske, H. H. Yang, N. Sherwani. "Integrated floorplanning with buffer/channel insertion for bus-based microprocessor designs", ISPD-02, pp. 56-61, 2002.

[5] X. Tang, R. Tian and D. F. Wong. "Fast evaluation of sequence pair in block placement by longest common subsequence computation", DATE-00, pp. 106-111, 2000.

[6] X. Tang and D. F. Wong. "FAST-SP: A fast algorithm for block placement based sequence pair", ASPDAC-01, pp. 521-526, 2001.

[7] X. Tang and D. F. Wong. "Floorplanning with alignment and performance constraints", DAC-02, pp. 848-853, Jun 2002.

[8] F. Y. Young, C. N. Chu and M. L. Ho. "A unified method to handle different kinds of placement constraints in floorplan design", Proceedings of the 15th International Conference on VLSI Design, Bangalore, India, pp. 661-667, Jan 2002.