RC23190 (W0404-114) April 21, 2004
Electrical Engineering

IBM Research Report

Relocate White Space in Floorplanning

Xiaoping Tang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Allocate White Space in Floorplanning

Abstract

Most existing floorplanning algorithms compact blocks to the left and bottom. Although the compacting obtains an optimal area, it may not be good to meet other objectives such as minimizing total wire length, routing congestion or buffer allocation. In this paper, we propose a new method to allocate white space in floorplanning for minimizing total wire length. Our method is based on min-cost flow implementation, and guarantees to obtain the minimum of total wire length for a given floorplan representation. We also show that the method can be easily extended to handle constraints such as fixed-frame (fixed area), boundary pins, pre-placed blocks, boundary blocks, range placement, alignment and abutment, rectilinear blocks, cluster placement, and bounded net delay, without lose of optimality. The algorithm is so efficient in that it finishes in less than 0.5 seconds for all MCNC benchmarks of block placement. Thus it can be either integrated into each step in simulated annealing or applied to post-floorplanning (refine floorplanning). It is also very effective. Experimental results show we can further improve 4.8% of wire length even on very compact floorplans.

1. Introduction

Floorplanning is to decide the positions of circuit blocks or IP blocks on a chip subject to various objectives. It is the early stage of physical design and determines the overall chip performance. Due to the enormous complexity of VLSI design with continuous scaling-down of technology, a hierarchical approach is needed for the circuit design in order to reduce runtime and improve solution quality. Also, IP (module reuse) based design methodology becomes widely adopted. This trend makes floorplanning even more important.

Floorplan can be classified into two categories, slicing and non-slicing. For slicing structure, there are binary tree[15] and normalized Polish expression[19]. For non-slicing structure, many representations are invented recently, such as topology representation (BSG[14], sequence pair[13], TCG[11]), packing representation(O-tree[7], B*-tree[5]), and mosaic representation (CBL[8], Q-sequence[16], twin binary tree[21], twin binary sequence[22]). All of these algorithms compact blocks to the left and bottom subject to the given topological constraints. Recently, additional constraints are addressed in floorplanning, such as fixed frame[17, 1], alignment and performance (bounded net delay)[18], buffer planning in floorplanning[12], etc.. Again, inside the approaches, the floorplan is compacted to lower-left (or upper-right) corner and then evaluated. In general, compacting implies minimum of area. However, it may be sub-optimal for other objectives, such as minimizing wire length, routing congestion, and buffer allocation. As we can see, even with the same minimum area and the same topology, there exist lots of different packings that have different values on other objectives. We illustrate the problem by a simple example in Figure 1.

We observe that in floorplanning and placement, minimizing total wire length is first-order objective. If a floorplanner/placer can minimize total wire length very well, then there is much freedom and space to consider and tradeoff other concerns such as routability and timing. Thus in the paper, we study the problem of minimizing total wire length in floorplanning, while leaving the consideration of routability and timing optimization (including buffer insertion) for future work.

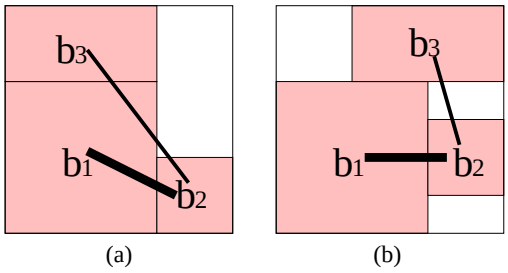

Figure 1: (a) The floorplan compacting blocks to the lower-left corner. However, the wire length is not optimal. (b) The floorplan with optimal wire length, which has the same topology but different allocation of white space. The dimensions for the 3 blocks are: $b_1(4\times4)$, $b_2(2\times2)$, $b_3(4\times2)$.

Most floorplanning algorithms use simulated annealing to search for an optimal floorplan. The implementation of simulated annealing scheme relies on a floorplan representation where a neighbor solution is generated and examined by perturbing the representation. In the paper, we use sequence pair as representation. The reason we pick sequence pair is that the evaluation of x and y coordinates can be done independently in sequence pair representation, which may cause our approach faster. However, our approach is not limited to sequence pair representation. For any floorplan represented by any other presentation, we can derive a constraint graph and thus apply the approach to allocate white space for minimizing total wire length.

Our approach is based on min-cost flow implementation, and guarantees to obtain the minimum of total wire length for a given floorplan representation. We also show that the approach is capable of handling various constraints such as fixed-frame (fixed area), boundary pins, pre-placed blocks, boundary blocks, range placement, alignment and abutment, rectilinear blocks, cluster placement, and bounded net delay, without lose of optimality. The algorithm is so efficient in that it finishes in less than 0.5 seconds for all MCNC benchmarks of block placement. Thus it can be either integrated into each step in simulated annealing or applied to post-floorplanning (refine floorplanning). It is also very effective. Experimental results show we can further improve 4.8% of wire length even on very compact floorplans. It is noted that researchers have studied the problem of allocating white space in placement for various objectives[9, 2, 4]. These methods are heuristics in term of minimizing wire length. Our approach optimally minimizes wire length for a given floorplan, and may be applicable to placement (behaving as a post-placement step, which is left as future work).

The rest of the paper is organized as follows. Section 2 briefly reviews sequence pair and constraint graph construction to evaluate a sequence pair. Section 3 formulates the problem of allocat-

ing white space to minimize total wire length, and presents a min-cost flow based approach to solve it. The capabilities of handling various constraints such as fixed-frame, boundary pins, pre-placed block, boundary block, range placement, alignment and abutment, rectilinear block, cluster placement, bounded net delay, etc., are discussed in Section 4. Experimental results are reported in Section 5, followed by concluding remarks in Section 6.

2. Preliminary

A sequence pair is a pair of sequences of n elements representing a list of n blocks. The two sequences specify the geometric relations (such as left-of, right-of, below, above) between each pair of blocks as follows:

$$(...b_i...b_j..., ...b_i...b_j...) \Rightarrow b_i \text{ is to the left of } b_j \quad (1)$$

$$(...b_j...b_i..., ...b_i...b_j...) \Rightarrow b_i \text{ is below } b_j \quad (2)$$

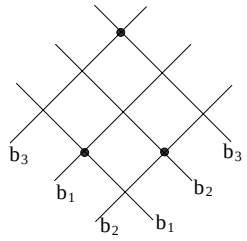

Figure 2: Sequence pair (b_3 b_1 b_2, b_1 b_2 b_3) specifies an oblique grid.

The sequence pair structure can be shown as an oblique grid (refer to Fig 2). The original paper which proposed sequence pair [13] presented an algorithm to translate a sequence pair to a placement by constructing two constraint graphs, G_h and G_v. Both G_h and G_v have $n+2$ vertices representing n blocks plus source node and sink node (representing boundaries). G_h has a directed edge (b_i, b_j) if block b_i is to the left of block b_j. Similarly, if block b_i is below block b_j, G_v has the corresponding directed edge (b_i, b_j). For any pair of blocks (e.g. b_i, b_j), there exists exactly one edge connecting the two nodes either in G_h or in G_v. Both G_h and G_v are vertex weighted, directed, acyclic graphs. The weights in G_h represent the widths of blocks, and the weights in G_v represent the heights of blocks. Given that the coordinates of a block are the coordinates of the lower-left corner of the block, a longest path algorithm can be applied to determine the coordinates of each block and the total width and height of the bounding box. As an example, the sequence pair specifying the placement in Figure 1 is (b_3 b_1 b_2, b_1 b_2 b_3).

3. Problem and Solution

Sequence pair specifies the topological relation between blocks. Given a sequence pair, previous algorithm compacts blocks to lower-left corner to minimize area. Even with the same minimum area, there exist different placements of blocks satisfying the topological constraint imposed by the sequence pair. It is very common that white space exists even in the floorplan packed to minimum area. The problem is to find a packing that minimize the total wire length, as defined as follows:

Problem 1. *Given a sequence pair (X,Y) with a set of m macro blocks B=$\{b_1, b_2, ..., b_m\}$ where $w_i \times h_i$ specifies the dimension of block b_i (w_i: width, h_i: height), and a set of nets $N = \{N_1, N_2, ..., N_n\}$ where $N_i, i = 1, 2, ..., n$ describes the connection between blocks, find a placement of blocks B satisfying the topological relation imposed by the sequence pair, such that the total wire length*

$$\sum_{i=1}^{n} \lambda_i W(N_i)$$

is minimized where $W(N_i)$ denotes the wire length of net N_i and λ_i is its weight.

Without loss of generality, we assume that all λ_i are integers. In the following we use x_i/y_i to denote the x/y coordinate of block b_i referring to the lower-left corner of the block. For simple representation and easy understanding, we assume all pins are locating in the center of the block. Actually as we can see later, our approach has no restriction that pins should be in the center of the block. It is common to use half perimeter of bounding box as an estimate of wire length for a net. Let us consider a net N_i connecting a set of z blocks $\{b_{i_1}, b_{i_2}, ..., b_{i_z}\}$, and use $(L_i, L'_i : R_i, R'_i)$ as its bounding box where (L_i, L'_i) and (R_i, R'_i) refer to lower-left and upper-right corner of the bounding box respectively. Thus we have $\forall j \in \{1, 2, ..., z\}$

$$L_i \leq x_{i_j} + w_{i_j}/2 \quad (3)$$

$$R_i \geq x_{i_j} + w_{i_j}/2 \quad (4)$$

$$L'_i \leq y_{i_j} + h_{i_j}/2 \quad (5)$$

$$R'_i \geq y_{i_j} + h_{i_j}/2 \quad (6)$$

Note that the coordinate $(x_{i_j} + w_{i_j}/2, y_{i_j} + h_{i_j}/2)$ is the center of the block b_{i_j} where pin is locating. When pin is not at the center, we can use the actual pin location to substitute the coordinate. In addition, the geometric constraint imposed by sequence pair can be written as follows:

$$(...b_i...b_j..., ...b_i...b_j...) \Rightarrow x_i + w_i \leq x_j \quad (7)$$

$$(...b_j...b_i..., ...b_i...b_j...) \Rightarrow y_i + h_i \leq y_j \quad (8)$$

Thus the problem can be stated as:

$$\min \sum_{i=1}^{n} \lambda_i (R_i - L_i + R'_i - L'_i) \quad (9)$$

subject to the set of constraints as stated in (3) (4) (5) (6) (7) (8). Since in sequence pair representation, the evaluation of x and y coordinates can be done independently, the problem can be decoupled into two subproblems:

$$\min \sum_{i=1}^{n} \lambda_i (R_i - L_i) \quad (10)$$

subject to the set of constraints as stated in (3) (4) (7), and

$$\min \sum_{i=1}^{n} \lambda_i (R'_i - L'_i) \quad (11)$$

subject to the set of constraints as stated in (5) (6) (8). The problems (10) and (11) can be solved separately. As we can see, all of the three problems, (9), (10) and (11), are linear programming. However, each of the problems has special property that all constraints are difference constraints[3]. Thus its dual problem is a min-cost flow problem, since in the constraint matrix of the dual problem, each column has exactly one "1" and "-1". Let us first consider the problem (10). We can construct a network graph (called horizontal network graph) $G_H = (V_H, E_H)$ as follows.

1. $V_H = \{s, t, x_1, x_2, ..., x_m, L_1, R_1, L_2, R_2, ..., L_n, R_n\}$, where s is the source node, t is the sink node, x_i represents the x coordinate of block b_i, and L_i and R_i represent the left and right boundary of bounding box of net N_i as denoted above.

2. $E_H = \{(s,v)|v = R_1, R_2, ..., R_n\} \cup \{(x_i, x_j)|$block b_i is to the right of block $b_j\} \cup \{(R_i, x_j), (x_j, L_i)|$net N_i connects to block $b_j\} \cup \{(u,t)|u = L_1, L_2, ..., L_n\}$, where (s,v) is the edge from source to right boundary of bounding box, (x_i, x_j) is the edge imposed by the sequence pair as in constraint (7), (R_i, x_j) is the edge imposed by net connection as in constraint (4), (x_j, L_i) is the edge imposed by net connection as in constraint (3), and (u,t) is the edge from left boundary of bounding box to sink.

3. Edge Capacity: $U_H(s,R_i) = U_H(L_i,t) = \lambda_i, \forall i \in \{1,2,...,n\}$; for any other edge $e \in E_H$, $U_H(e)$ is unlimited.

4. Cost Function: $C_H(s,v) = 0, C_H(u,t) = 0, C_H(x_i,x_j) = -w_j$, and $C_H(R_i,x_j) = C_H(x_j,L_i) = -w_j/2$.

It should be noted that the subgraph, which contains only the vertices $x_i, i = 1,2,...m$ and the edges (x_i,x_j) imposed by sequence pair, is similar to the horizontal constraint graph mentioned in [13]. The difference is that the direction of edges is inverted and the edge cost is negative. Thereafter, in [13] a longest path algorithm is applied to compute the positions of blocks, while in the paper we shall use min-cost flow algorithm in the sense of shortest path. It should also be noted that the transitive edges on the subgraph can be safely omitted, which will speed up the computation considerably.

Thus we compute the min-cost flow of amount $\sum_{i=1}^{n} \lambda_i$ on the graph G_H, which solves the dual problem. Our goal is to compute the positions of blocks subject to the constraints and minimize the total wire length (the primal problem), which can be done as follows. We first compute the residual graph derived from the min-cost flow. Then a shortest path algorithm applied on the residual graph would give the positions for all blocks. If necessary, a common source node connecting to all other nodes can be added to the residual graphs for shortest path computation.

Analogously, we can construct another network graph and solve the problem (11) by min-cost flow approach. The graph (called vertical network graph) $G_V = (V_V, E_V)$ is constructed as follows.

1. $V_V = \{s,t,y_1,y_2,...,y_m,L'_1,R'_1,L'_2,R'_2,...,L'_n,R'_n\}$, where s is the source node, t is the sink node, y_i represents the y coordinate of block b_i, and L'_i and R'_i represent the lower and upper boundary of bounding box of net N_i as denoted above.

2. $E_V = \{(s,v)|v = R'_1, R'_2, ..., R'_n\} \cup \{(y_i, y_j)|$block b_i is above block $b_j\} \cup \{(R'_i, y_j), (y_j, L'_i)|$net N_i connects to block $b_j\} \cup \{(u,t)|u = L'_1, L'_2, ..., L'_n\}$, where (s,v) is the edge from source to upper boundary of bounding box, (y_i, y_j) is the edge imposed by the sequence pair as in constraint (8), (R'_i, y_j) is the edge imposed by net connection as in constraint (6), (y_j, L'_i) is the edge imposed by net connection as in constraint (5), and (u,t) is the edge from lower boundary of bounding box to sink.

3. Edge Capacity: $U_V(s,R'_i) = U_V(L'_i,t) = \lambda_i, \forall i \in \{1,2,...,n\}$; for any other edge $e \in E_V$, $U_V(e)$ is unlimited.

4. Cost Function: $C_V(s,v) = 0, C_V(u,t) = 0, C_V(y_i,y_j) = -h_j$, and $C_V(R'_i,y_j) = C_V(y_j,L'_i) = -h_j/2$.

We use the example as shown in Figure 1 to illustrate the approach. The input of the problem is: sequence pair (b_3 b_1 b_2, b_1 b_2 b_3) with 3 blocks, and nets $N_1 = \{b_1, b_2\}$ with weight $\lambda_1 = 2$, $N_2 = \{b_2, b_3\}$ with weight $\lambda_2 = 1$. Then the problem (10) to minimize wire length in x dimension can stated as follows:

$$\min\{2(R_1 - L_1) + (R_2 - L_2)\}$$

subject to

$$\begin{aligned}
x_1 + 4 &\leq x_2 \\
x_1 + 2 &\geq L_1 \\
x_1 + 2 &\leq R_1 \\
x_2 + 1 &\geq L_1 \\
x_2 + 1 &\leq R_1 \\
x_2 + 1 &\geq L_2 \\
x_2 + 1 &\leq R_2 \\
x_3 + 2 &\geq L_2 \\
x_3 + 2 &\leq R_2
\end{aligned}$$

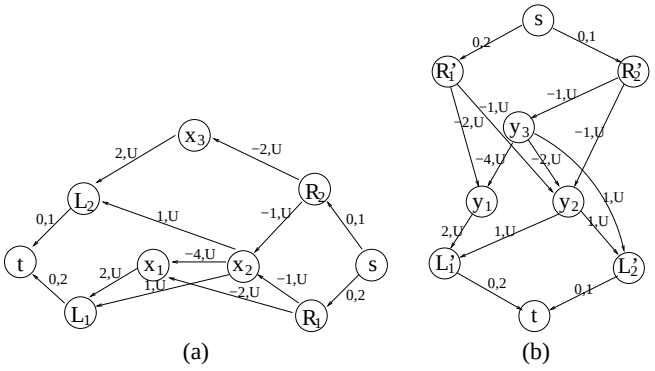

Figure 3: (a) The horizontal network graph. (b) The vertical network graph. The pair of numbers, "c,u", on the edge represent cost and capacity respectively, and "U" means unlimited capacity (the same meaning on the graphs that follows).

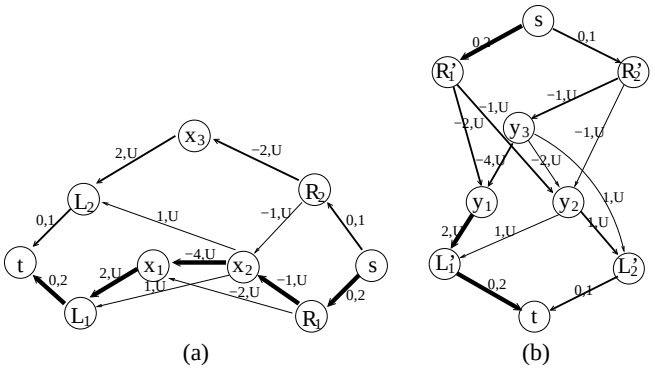

Figure 4: (a) The min-cost flow on the horizontal network graph. (b) The min-cost flow on the vertical network graph. The highlighted lines represent flows. Their widths are proportional to the amount of flow.

Then this can be transformed to min-cost flow problem in the network graph G_H as shown in Figure 3(a). Similarly, the problem (11) to minimize wire length in y dimension is transformed to min-cost flow problem in the network graph G_V as shown in Figure 3(b). Then we compute the min-cost flow of amount: 3 (because $\lambda_1 + \lambda_2 = 3$) on the two graphs, G_H and G_V. The results are illustrated in Figure 4(a) and Figure 4(b) respectively. Based on the flow results, we derive the residual graphs of G_H and G_V, as shown in Figure 5(a) and Figure 5(b) respectively. Thus we apply shortest path algorithm on the residual graphs to compute the positions

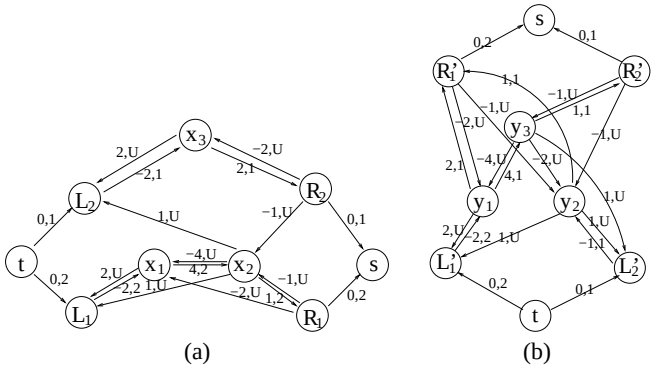

Figure 5: (a) The horizontal residual graph. (b) The vertical residual graph.

of blocks by adding a common source node connecting all other nodes. Thus the results are: $x_1 = -5$, $x_2 = -1$, $x_3 = -2$, $y_1 = -5$, $y_2 = -4$, and $y_3 = -1$. The placement with minimum wire length is shown in Figure 6. The overall approach is summarized as follows.

Algorithm Min-wire
1. Construct the network graphs G_H and G_V
2. Apply min-cost flow algorithm on G_H and G_V
3. Derive the residual graphs of G_H and G_V
4. Apply shortest path algorithm on residual graphs to compute positions of blocks

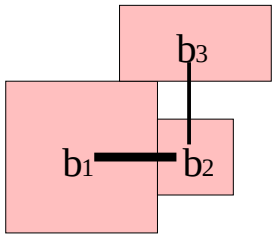

Figure 6: The placement with minimum wire length.

The min-cost flow based algorithm, Min-wire, optimally solves the problem of minimizing total wire length, as stated in the following theorem.

Theorem 1. *The algorithm, Min-wire, generates a placement of all blocks such that the total wire length is minimized optimally for the given sequence pair.*

The complexity of the algorithm, Min-wire, is determined by min-cost flow, since other steps are smaller portions compared to min-cost flow algorithm. Finding a min-cost flow in a network is a classical problem for which several polynomial-time optimal algorithms are available [3]. The number of vertices in either G_H or G_V is $O(m+n)$ where m is the number of blocks and n is the number of nets. The number of edges on the subgraph, which contains only the vertices representing blocks and the edges between, is $O(m \log m)$ on average[10]. The rest of edges includes the edges introduced by net connections, and the edges incident from/to source/sink. The number of edges incident from source and to sink is $O(n)$. The edges introduced by net connections in the graph is proportional to the number of pins in all nets. Typically in practice, we can assume that the number of pins is a constant in a net. Thus the number of edges introduced by net connections is typically $O(n)$. In total, the number of edges is $O(m \log m + n)$. Therefore, if we adopt Orlin's algorithm in [3] to compute the min-cost flow, the time complexity of the algorithm Min-wire is typically $O(|E|\log|V|(|E|+|V|\log|V|)) = O((m\log m+n)\log(m+n)(m\log m+n+(m+n)\log(m+n)))$. Practically, we can assume that net weight λ_i is $O(1)$ (for example, 1-10), which is true in most applications. We observe that too large weight is unnecessary in actual applications. When net weight is beyond some threshold, it behaves the same in minimizing wire length. Then we can apply successive shortest path augmenting algorithm in computing min-cost flow[3]. Thus, the complexity is $O(nS(m,n))$ where $S(m,n)$ denotes the time taken to solve a shortest path problem. If we associate each node with an adjusted weight to eliminate negative cost[3], then $S(m,n)$ is the complexity of Dijkstra algorithm. Finally the complexity is $O(n(|V|\log|V|+|E|)) = O(n(m\log m+n+(m+n)\log(m+n)))$.

4. Discussion of Capabilities

In the section, we discuss the capabilities of the approach in handling various constraints.

4.1 Fixed-frame

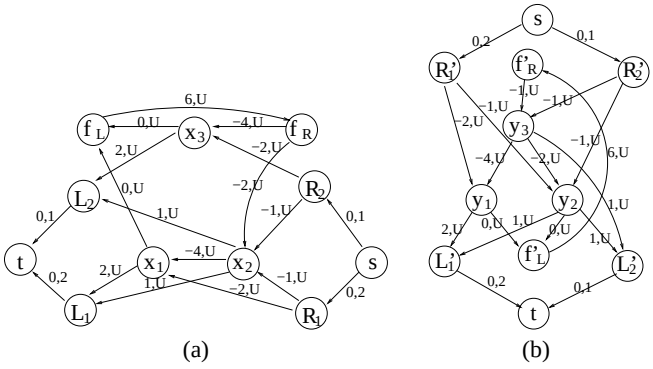

Figure 7: The modification of graphs to handle fixed-frame where the frame is 6×6. (a) The horizontal network graph. (b) The vertical network graph.

In some applications, floorplanning is confined in a given frame, $W \times H$, where W and H represent width and height respectively. In addition, if we still want to keep the minimum area in minimizing wire length, we can solve the problem with a frame of minimum area. When a frame is taken into account, we modify the graphs as follows. To horizontal network graph G_H, two nodes, f_L and f_R, are added where f_L and f_R represent the left and right boundary of the frame respectively. A set of edges are added, (f_R, x_i) with cost $-w_i$ and unlimited capacity, (x_i, f_L) with cost 0 and unlimited capacity, $i = 1, 2, \ldots, m$, and (f_L, f_R) with cost W and unlimited capacity. Again, the transitive edges can be omitted. Similarly, two nodes, f'_L and f'_R representing the lower and upper boundary of the frame respectively, and the corresponding edges are added to vertical network graph G_V. Figure 7 illustrates the two modified graphs for the example above. The frame is 6×6 (minimum area). Thus the algorithm Min-wire can still be applied to minimize the total wire length and place blocks in the given frame. Note that the node f_R (f'_R) will act as the source node in the step of shortest path computation in G_H (G_V) to obtain the positions of blocks. Figure 1(b) actually gives the optimal placement within the frame.

4.2 Boundary Pin

Usually there exist IO nets which connect to pins on the boundary of frame (boundary pins). Let us consider a net N_i connects a boundary pin at location (p_x, p_y). Thus $L_i \le p_x \le R_i$ and $L'_i \le p_y \le R'_i$. Assume the frame is $W \times H$. Then equivalently, $L_i - f_R \le p_x - W \le R_i - f_R$ and $L'_i - f'_R \le p_y - H \le R'_i - f'_R$. As a result, we add two edges, (f_R, L_i) (with cost $p_x - W$ and unlimited capacity) and (R_i, f_R) (with cost $W - p_x$ and unlimited capacity), to graph G_H, and add two edges, (f'_R, L'_i) (with cost $p_y - H$ and unlimited capacity) and (R'_i, f'_R) (with cost $H - p_y$ and unlimited capacity), to graph G_V. Thus the algorithm Min-wire can be applied. In this way, fixed pin can be handled where the pin location may not be on the boundary of frame.

4.3 Pre-placed and Boundary Blocks

In the situation where some blocks are to be placed at fixed location or on the boundary of the frame, the algorithm can still apply by adding additional edges to graphs. For example, a block b_i is placed at a location (l_x, l_y), i.e. $x_i = l_x$ and $y_i = l_y$. Thus $x_i - f_R = l_x - W$ and $y_i - f'_R = l_y - H$. Equivalently, $x_i - f_R \le l_x - W$, $x_i - f_R \ge l_x - W$, $y_i - f'_R \le l_y - H$, and $y_i - f'_R \ge l_y - H$. These are transformed to edges (f_R, x_i) and (x_i, f_R) on graph G_H, and edges (f'_R, y_i) and (y_i, f'_R) on graph G_V. Boundary blocks can be handled similarly in the sense that boundary blocks fix locations in x or y coordinate.

4.4 Range Placement

Range constraint specifies that a block is to be placed within a given range. Pre-placed constraint is a special case of range constraint. Similarly, we can add additional edges to graph G_H and G_V to enforce the computation of position in algorithm Min-wire such that the block is placed within the range.

4.5 Alignment and Abutment

Alignment constraint specifies several blocks to be aligned in a row within a range[18]. It can be transformed to a set of difference constraints that keep the relative positions between them. Thus we can add additional edges to the graphs accordingly. Abutment is a special case of alignment.

4.6 Rectilinear Block

Rectilinear block is partitioned into a set of rectangular subblocks. Then a set of constraints are used to keep the relative positions, which can be transformed to the additional edges in the graphs accordingly[6].

4.7 Cluster Placement

It is useful in applications that several blocks are placed close to each other (cluster placement). In other words, the distance between any two of the blocks should not be too far away. This can be written as a set of constraints that specify the distance bound between any two of the blocks. Thus we can solve the problem by adding the corresponding edges to the graphs.

4.8 Bounded Net Delay

The approach is to minimize the total wire length, which can not guarantee bounded delay for critical nets. To address bounded net delay, we use a linear function in terms of distance to estimate delay. Although interconnect delay is quadratic in terms of wire length, with appropriate buffer insertions the actual delay is close to linear in terms of source-sink distance. In this way we convert bounded net delay into bounded net wire length. Thus as in [18], we impose constraints on the bounding box of the net, which results in the additional edges in the graphs accordingly.

4.9 Composite Cost Function

We have talked about fixed-frame constraint that blocks are confined within a given frame. Actually, the method is an exact algorithm to optimize the composite cost function of area and wire length:

$$\min\{\alpha(W+H) + \sum_{i=1}^{n} \lambda_i W_i\}$$

Note that existing methods can only minimize area, and use compacted blocks' locations to compute the cost of wire length. The modification to the graphs, G_H and G_V, is as follows.

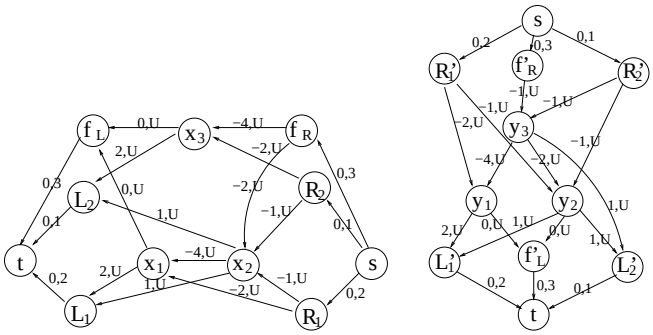

Figure 8: The modification of graphs to minimize composite cost where $\alpha = 3$. (a) The horizontal network graph. (b) The vertical network graph.

It is similar to the modification that handles fixed-frame constraints, except that there is no edge (f_L, f_R) on graph G_H and no edge (f'_L, f'_R) on graph G_V. Instead, we add edges, (s, f_R) with cost 0 and capacity α and (f_L, t) with cost 0 and capacity α, to graph G_H; and we add edges, (s, f'_R) with cost 0 and capacity α and (f'_L, t) with cost 0 and capacity α, to graph G_V. Figure 8 illustrates the two modified graphs for the example above to optimize the composite cost function, where $\alpha = 3$. Thus the min-cost flow based algorithm can be used to optimally minimize the composite cost.

We have the following necessary and sufficient condition with respect to all these constraints.

Theorem 2. *There exists a feasible placement that satisfies all these constraints if and only if there is no negative cycle in graphs G_H and G_V.*

Although the condition is similar to that in [6], there exist important differences. (i) The graphs are different. The graph in [6] contains only nodes representing blocks/subblocks. (ii) The approach in [6] operating on its graph thus does area packing only, while our approach can minimize both area and wire length. (iii) Longest path algorithm is used in [6], while main part of our algorithm is min-cost flow.

When a graph has a negative cycle with unlimited capacity, there does not exist min-cost flow. As we can see, in the graph G_H and G_V, the edges except the edges incident from source node or to sink node have unlimited capacity, and the edges incident from source node or to sink node can not be part of any cycle. Thus any negative cycle will have unlimited capacity. If there is no negative cycle, then the algorithm Min-wire can be used to compute a placement that satisfies all constraints and has the minimum of wire length.

5. Experimental Results

Table 1: Results of floorplanning in minimizing wire length.

circuit	block	net	no frame				min-area frame			
			wire(*mm*)		improve	time(s)	wire(*mm*)		improve	time(s)
			original	after			original	after		
apte	9	97	211.4	201.8	4.5%	0.05	405.6	402.3	0.8%	0.05
xerox	10	203	540.7	501.1	7.3%	0.09	543.2	493.7	9.1%	0.10
hp	11	83	137.9	133.0	3.6%	0.01	228.7	221.7	3.0%	0.05
ami33	33	123	63.4	57.4	9.4%	0.07	62.1	60.6	2.5%	0.07
ami49	49	408	816.3	773.7	5.2%	0.40	767.6	747.6	2.6%	0.47

We have implemented the algorithm and integrated with the floorplanner, FAST-SP[17]. Assuming that $\lambda_i = O(1)$, we use successive shortest path augmenting algorithm in min-cost flow computation. The test problems are derived from MCNC benchmarks for block placement.

We first use FAST-SP to obtain a good floorplan candidate, and then apply the algorithm Min-wire to further optimize wire length. For all tests, we use the center of block as pin's location. In the first set of experiments, there is no frame constraints. The second set of experiments has a fixed frame of minimum area. Table 1 lists the experimental results for minimizing wire length, where all blocks are hard blocks. The experiments were carried out on a Pentium 4 Mobile(2.4Ghz). As we can see, the algorithm is very efficient in that it takes less than 0.5 seconds for all of the benchmarks. Thus it can be integrated into each step in simulated annealing to get a better result of wire length. Here in the table, we only give the result of floorplanning refinement (post-floorplanning). It is also very effective in that it can further improve 4.8% of wire length on average even on very compact floorplans. As illustrations, Figure 9 and 10 display the placement results of original and after optimization for ami33 and ami49 respectively.

6. Concluding Remarks

In the paper, we have presented a novel method to allocate white space in floorplanning. The method optimally distributes white space among blocks and guarantees to obtain the minimum of total wire length for a given floorplan representation. It is also an exact algorithm to optimize the composite cost function of area and wire length: $\min\{\alpha(W+H)+\sum_{i=1}^{n}\lambda_i W_i\}$. We have also shown that the method can handle various constraints such as fix-frame, boundary pins, pre-placed blocks, boundary blocks, range placement, alignment and abutment, rectilinear blocks, cluster placement, and bounded net delay, without lose of optimality. Experimental results show it is very efficient and effective in that it finishes in less than 0.5 seconds for all of the MCNC benchmarks and improves 4.8% of wire length even on very compact floorplans. Thus it provides an ideal way to refine floorplanning (post-floorplanning), and can be integrated into each step in simulated annealing as well. The future work is to extend the method to consider routing congestion and buffer insertion in floorplanning and to apply it in placement.

7. References

[1] S.N. Adya and I.L. Markov. "Fixed-outline floorplanning through better local search", ICCD-01, pp. 328-334, 2001.

[2] S.N. Adya, I.L. Markov, and P.G. Villarrubia. "On whitespace and stability in mixed-size placement and physical synthesis", ICCAD'03, pp. 311-317, 2003.

[3] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. Network Flows, Prentice Hall, 1993.

[4] C.J. Alpert, G.J. Nam, and P.G. Villarrubia. "Free space management for cut-based placement", ICCAD'02, pp. 746-751, 2002.

[5] Y.C. Chang, Y.W. Chang, G.M. Wu, and S.W. Wu. " B*-trees: a new representation for non-slicing floorplans", DAC-2000, pp. 458-463, 2000.

[6] K. Fujiyoshi, and H. Murata. "Arbitrary convex and concave rectilinear block packing using sequence pair", ISPD-99, pp. 103-110, 1999.

[7] P.N. Guo, C.K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplans and its applications", DAC-99, pp. 268-273, 1999.

[8] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.K. Cheng, and J. Gu. "Corner block list: an effective and efficient topological representation of non-slicing floorplan", ICCAD-00, pp. 8-12, 2000.

[9] A.B. Kahng, P. Tuckler, and A. Zelikovsky. "Optimization of linear placements for wirelength minimization with free sites", ASP-DAC'99, pp. 241-244, 1999.

[10] C. Lin. "Incremental mixed-signal layout generation concepts", phd thesis, 2002.

[11] J.M. Lin and Y.W. Chang. "TCG: a transitive closure graph-based representation for non-slicing floorplans", DAC-01, pp. 764-769, 2001.

[12] Y. Ma, X. Hong, S. Dong,S. Chen, Y. Cai, C.K. Cheng, and J. Gu, "An integrated floorplanning with an efficient buffer planning algorithm", ISPD'03, pp. 136-142, 2003.

[13] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. "VLSI module placement based on rectangle-packing by the sequence pair", *IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems*, vol. 15:12, pp. 1518-1524, 1996.

[14] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani. "Module placement on BSG-structure and IC layout applications", ICCAD-96, pp. 484-491, 1996.

[15] R.H.J.M. Otten. "Automatic floorplan design", DAC-82, pp. 261-267, 1982.

[16] K. Sakanushi and Y. Kajitani. "The quarter-state sequence (Q-sequence) to represent the floorplan and applications to layout optimization", IEEE APCCAS, pp. 829-832, 2000.

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

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

[19] D.F. Wong and C.L. Liu. "A new algorithm for floorplan design", DAC-86, pp. 101-107, 1986.

[20] X. Yang, B.K. Choi and M. Sarrafzadeh. "Routability driven white space allocation for fixed-die standard cell placement", ISPD'02, pp. 42-50, 2002.

[21] B. Yao, H. Chen, C.K. Cheng, and R. Graham. "Revisiting floorplan representation", ISPD-01, pp. 138-143, 2001.

[22] F.Y. Young, C.N. Chu, and Z.C. Shen. "Twin binary sequences: a non-redundant representation for general non-slicing floorplan", ISPD-02, pp. 196-201, 2002.

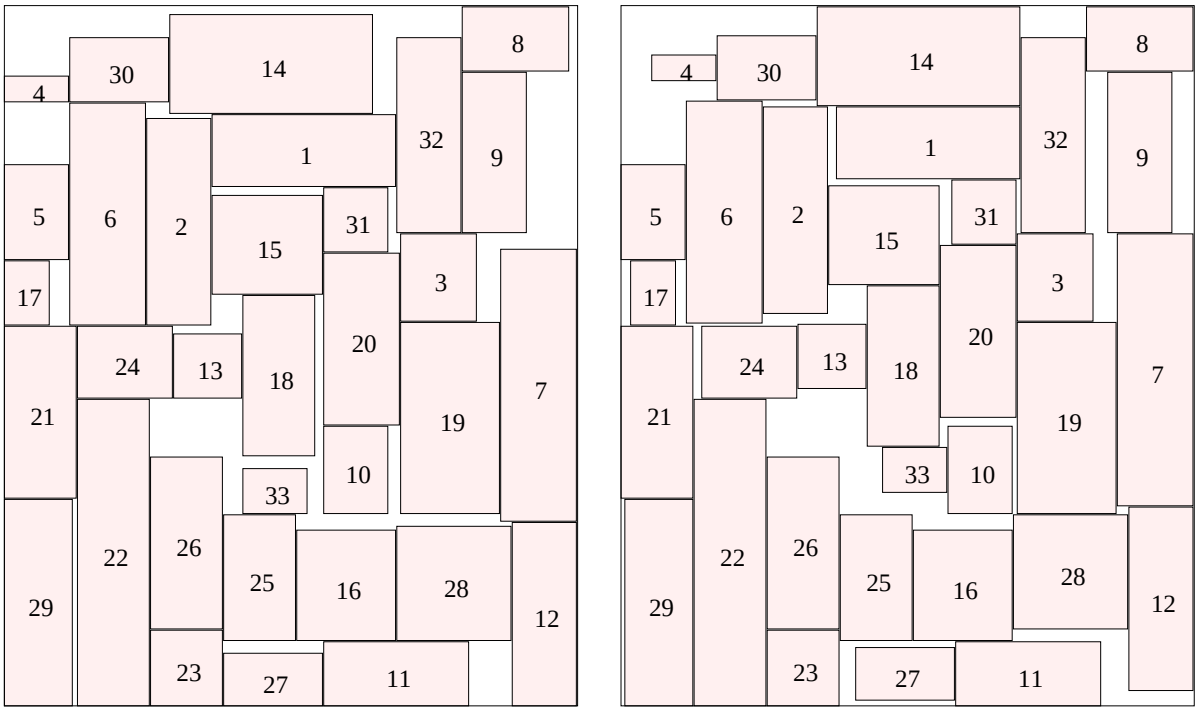

Figure 9: (a) The original ami33 placement. (b) The new placement result when minimizing wire length in the same frame.

Figure 10: (a) The original ami49 placement. (b) The new placement result when minimizing wire length in the same frame.