Floorplanning with Alignment and Performance Constraints [*]

Xiaoping Tang[†‡] and D.F. Wong[†]
[†]University of Texas at Austin, Austin, TX 78712
[‡]Silicon Perspective, A Cadence Company, Santa Clara, CA 95054
{tang, wong}@cs.utexas.edu

ABSTRACT

In this paper, we present a floorplanning algorithm based on sequence pair representation. Our floorplanner has the following important features: 1) It is explicitly designed for fixed-frame floorplanning, which is different from traditional well-researched min-area floorplanning. Moreover, we also show that it can be adapted to minimize total area. 2) It addresses the problem of handling alignment constraint which arises in bus structure. 3) It deals with performance constraint such as bounded net delay, while many existing floorplanners just minimize total wire length. 4) More importantly, even with all these constraints the algorithm is very fast in that it evaluates the feasibility of a sequence pair and translates to a floorplan in $O(n \log \log n)$ time typically where n is the number of blocks and the number of constrained blocks is $O(n)$, which is significantly faster than the $O(n^3)$ method operating on constraint graph. Our algorithm is based on computing the longest common subsequence of a pair of weighted sequences. Experimental results on MCNC benchmark for block placement show the promise of the method.

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

Keywords

longest common subsequence, floorplanning, sequence pair

1. INTRODUCTION

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 used. This trend makes floorplanning more and more important. 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 design and it determines the overall chip performance. Most floorplanning algorithms use simulated annealing to search for an optimal solution. The implementation of simulated annealing scheme relies on a floorplan representation where a neighbor solution is generated and examined by perturbing the representation (called 'move'). As a result, many researchers explore in this area [2, 5, 6, 10, 11, 12, 13, 14, 17].

Floorplans can be classified into two categories, slicing and non-slicing. For slicing structure, Otten[13] proposed a binary tree representation, and later Wong and Liu[17] presented a normalized Polish expression to represent a slicing floorplan. Slicing structure has advantages, but typical floorplan is non-slicing. Lai and Wong[9] showed that based on 1-D compaction, slicing floorplan can be transformed to non-slicing floorplan without missing any min-area floorplan.

Since mid-1990s, several representations for non-slicing structure are invented, such as BSG[11], sequence pair[10], O-tree[5], B*-tree[2], CBL[6], and Q-sequence[14]. They can be further categorized into three classes. BSG and sequence pair specify the topological relative positions between blocks such as left-of, right-of, below, and above. These two can represent general floorplan. Block placement can be obtained by compacting blocks subject to relative positions. O-tree and B*-tree describe 1-D relative positions (horizontal adjacency) among blocks. The final placement is obtained by compacting blocks to the left and bottom. Although O-tree and B*-tree have smaller solution space, they are dimension dependent, i.e., geometric relation between blocks can not be obtained directly from representation. CBL and Q-sequence are two coding schemes for non-slicing floorplans called Mosaic[6] where a chip is dissected into n rectangular rooms. However, they can not represent some general floorplan unless empty rooms are inserted purposely.

Among these representations, some have larger solution space implying more redundancy, and some have smaller solution space implying less redundancy. All floorplanning algorithms based on the representations use simulated annealing. One observation we made is that redundancy is not a major issue in simulated annealing. Instead, the effectiveness of simulated annealing is more related to the diameter of solution space (diameter is the maximum distance between any two solutions where the distance between two solutions is the minimum number of moves transforming one to the other), how moves (neighbor solution) are generated, the complexity of evaluation of each representation (i.e. transforming a representation to a floorplan), and the cooling schedule in annealing process. Among the evaluations of these representations, some require quadratic runtime, some require linear runtime, and some require runtime between linear and quadratic. Another observation we made is that the difference in complexity could be nullified by implementation overhead and annealing process. Instead, the hidden constant in implementation complexity plays a major role in overall runtime. Recently, Tang et al sped up the original $O(n^2)$-time evaluation algorithm for sequence pair to $O(n \log n)$ in [15], and later further to $O(n \log \log n)$ in [16]. The experimental results in [16] show that it outperforms existing floorplanning algorithms, which implies that sequence pair is a competitive representation. In the paper, we use the sequence pair representation.

Kahng[7] suggested a fixed-frame floorplanning with no dead space. Zero dead space usually requires non-rectangular blocks. However, arbitrary block shape is undesirable in floorplanning stage. Instead the rectilinear polygon with bounded number of edges is more preferable. Of course, post-processing can be applied for floorplanning to entail arbitrary block shape. Actually, most existing algorithms handle rectilinear block by partitioning into a set

[*]This work was partially supported by the National Science Foundation under grant CCR-9912390.

of rectangular subblocks. Traditional floorplanners minimize the area of bounding box, or use min-area as one of the objectives. A fixed-frame floorplanning algorithm was proposed in [16] to handle range and boundary constraints. Adya and Markov considered both frame-free and fixed-frame floorplanning optimizations in [1]. In the paper, we explicitly design the strategy for fixed-frame floorplanning. To be consistent with traditional floorplanning, we also show that it can minimize area by shrinking the frame.

With bus structures or pipelines, it is better that the blocks are aligned in a row, abutting with each other (alignment constraint). The blocks can be aligned horizontally or vertically. Alignment constraint specifies that several blocks are aligned in a row with a range. It is different from abutment constraint because several abutted blocks may not be aligned. It is also different from rectilinear shape because it has more freedom in positioning. The techniques operating on constraint graph [4] can be applied to alignment constraint, but the complexity is high ($O(n^3)$).

Many existing floorplanners minimize total wire length, where there is no distinguishing between critical and non-critical nets. Even if the total wire length is optimally minimized, there is no guarantee for critical nets. Timing has become a more and more important issue in deep submicron design. Bounded delay for critical nets should be considered in early floorplanning stage, so minimizing total wire length is no longer sufficient. In the paper, we refer to bounded net delay as performance constraint.

With all these constraints (fixed-frame, alignment, and performance) taken into account, it is clear that some sequence pair is not feasible. It can be shown that any placement that satisfies the given constraints is represented by a sequence pair. We first derive necessary conditions of feasible sequence pair satisfying alignment constraint in terms of the composition of the sequence pair. Then only the sequence pairs which satisfy the necessary conditions are evaluated, implying reduced solution space. We also derive a necessary and sufficient condition of feasible sequence pair with respect to all of the three constraints, fixed-frame, alignment, and performance. Moreover, a fast algorithm is designed to evaluate the feasibility of a sequence pair and translate to a floorplan in $O(n \log\log n)$ time typically. The algorithm is based on computing the longest common subsequence of a pair of weighted sequences. Finally, since the feasibility of a sequence pair depends on the actual dimension of blocks, it is impossible to search in solution space including only feasible sequence pairs. We use simulated annealing to search for optimal floorplan satisfying given constraints, where a novel cost function is used to unify the evaluation of feasible and infeasible sequence pairs. The algorithm is very fast. Our experimental results on MCNC benchmark for block placement show the promise of the method.

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:

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

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

The original paper which proposed sequence pair [10] 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. The evaluation of sequence pair takes $\Theta(n^2)$ time.

Tang et al[15] showed that the coordinates of blocks and the total width and height of floorplan can be obtained by computing longest common subsequence in terms of the two sequences. Given a sequence pair (X, Y), the total width of floorplan equals the length of the longest common subsequence of X and Y where weights are blocks' widths. Analogously, the total height of floorplan is determined by dealing with the longest common subsequence of X^R and Y where X^R is the reverse of X and weights are blocks' heights. The coordinates of a block are calculated as follows. Let (X, Y)=$(X_1 b X_2, Y_1 b Y_2)$ and $lcs(X, Y)$ denote the length of the longest common subsequence of X and Y. Then (X^R, Y) = $(X_2^R b X_1^R, Y_1 b Y_2)$. The x-coordinate of block b equals $lcs(X_1, Y_1)$ with blocks' widths as weights. The y-coordinate of block b is $lcs(X_2^R, Y_1)$ with blocks' heights as weights. In addition, all the computations of blocks' x/y coordinates can be integrated into a single longest common subsequence computation for a sequence pair.

3. ALIGNMENT CONSTRAINT

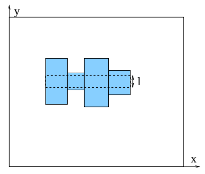

Figure 1: 4 blocks are aligned horizontally. l is the alignment range (e.g. bus width). The relative vertical positions are not fixed as long as the blocks are aligned within the given range l.

Alignment constraint specifies that several blocks are aligned in a range (e.g. bus width), abutting one by one horizontally or vertically. It is useful to facilitate data transfer in bus structure or a pipeline. As an example, Fig 1 illustrates that 4 blocks are aligned horizontally. Aligned blocks are abutted with each other in a row. But alignment is different from abutment in the sense that several abutted blocks may not be aligned . Moreover, it has more freedom in positioning than rectilinear shape because the relative positions are not fixed.

Alignment is classified into H-alignment and V-alignment according to the orientation, horizontal and vertical respectively. H-alignment can be formally defined as follows.

DEFINITION 1. **H-alignment***: Given a range l and k blocks, b_i, $i = 1,2,\ldots,k$, with dimension $w_i \times h_i$ and coordinates (x_i, y_i) referring to the lower-left corner, $i = 1,2,\ldots,k$ respectively, the k blocks are H-aligned iff $x_i + w_i = x_{i+1}$, $1 \le i \le k-1$ (abutting one by one) and $y_{max} + l \le y_i + h_i$, $1 \le i \le k$, where $y_{max} = max\{y_i | i = 1,2,\ldots,k\}$ (aligning horizontally).*

Similarly, we can define V-alignment.

DEFINITION 2. **V-alignment***: Given a range l and k blocks, b_i, $i = 1,2,\ldots,k$, with dimension $w_i \times h_i$ and coordinates (x_i, y_i), $i = 1,2,\ldots,k$ respectively, the k blocks are V-aligned iff $y_i + h_i = y_{i+1}$, $1 \le i \le k-1$ (abutting one by one) and $x_{max} + l \le x_i + w_i$, $1 \le i \le k$, where $x_{max} = max\{x_i | i = 1,2,\ldots,k\}$ (aligning vertically).*

In the following, we first discuss H-alignment. Then we show V-alignment can be handled similarly. If k blocks, b_i, $i = 1,2,\ldots,k$, are H-aligned, then in sequence pair representation, it must be like $(X,Y) = (\ldots b_1 \ldots b_2 \ldots\ldots b_i \ldots\ldots b_k \ldots, \ldots b_1 \ldots b_2 \ldots\ldots b_i \ldots\ldots b_k \ldots)$ maintaining the same order. Moreover, b_i must be **strictly ahead** of b_{i+1}. "Strictly ahead" is defined as follows.

DEFINITION 3. **strictly ahead***: Given 2 blocks, a and b, and two sequences $(X, Y) = (X_1 a X_2 b X_3, Y_1 a Y_2 b Y_3)$, a is strictly ahead of b in (X, Y) iff $lcs(X_2, Y_2) = 0$.*

Since we are dealing with blocks which only have non-zero dimension, $lcs(X_2, Y_2) = 0$ means no common elements between X_2 and Y_2. For example, in two sequences $(a\ b\ c\ d\ e, a\ d\ c\ b\ e)$, a

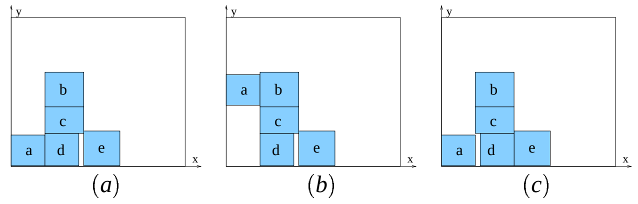

Figure 2: Given a sequence pair (X,Y)=(a b c d e, a d c b e), (a) Block a abuts with d (a is strictly ahead of d in (X,Y)); (b) a can be moved up to abut with b (a is strictly ahead of b in (X,Y)); (c) d can be moved right to abut with e since d is strictly ahead of e in (X,Y).

is strictly ahead of b, c or d, but not strictly ahead of e. "Strictly ahead" implies that the two blocks can possibly be abutted with each other, otherwise abutment is impossible. For example, as shown in Fig 2(a), block a abuts with d given that sequence pair be (a b c d e, a d c b e). By moving a up (aligning), a can abut with b (in Fig 2(b)). However, it can not abut with e. d can abut with e (in Fig 2(c)).

Thus we have the following necessary condition for H-alignment constraint in sequence pair.

THEOREM 1. **(adjacency-order)** *If k blocks, b_i, $i = 1,2,...,k$, are H-aligned, then the corresponding sequence pair representation must satisfy that $(X,Y) = (...b_1...b_2......b_i......b_k...,...b_1...b_2......b_i......b_k...)$ and b_i is strictly ahead of b_{i+1} in (X,Y), $1 \le i \le k-1$.*

If a sequence pair satisfies the above condition given in Theorem 1, the constrained blocks can possibly be abutted one by one in x direction and then be aligned in y direction (moving upward). Note that the alignment operation does not change the topological relation specified by sequence pair. Generally, the H-alignment operation is performed as follows:

$$y_{max} \leftarrow max\{y_i | i = 1,2,...,k\}$$
$$y_i \leftarrow max(y_i, y_{max} + l - h_i) \quad \forall i \in \{1,2,...,k\}$$

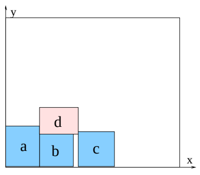

Figure 3: The sequence pair (a d b c, a b d c) satisfies the condition in Theorem 1, but the constrained blocks a,b and c can not be H-aligned.

Note that the condition in Theorem 1 is only necessary, but not sufficient. For example, as shown in Fig 3, the sequence pair (a d b c, a b d c) satisfies the condition, but the constrained blocks a,b and c can not be H-aligned because b and c are not tightly abutted (d is wider than b). We solve the problem by adjusting the x coordinate of the first block in an H-alignment constraint to make them tightly abutted . The adjustment is performed as follows: $x_1 \leftarrow x_k - \sum_{i=1}^{k-1} w_i$. And then we add a "dummy" block at the right as shown in Fig 4(a) and re-evaluate the sequence pair. If the constrained blocks can not be H-aligned, then the dummy block will be pushed out of frame as shown in Fig 4(b). Formally, the dummy block is determined as follows. Given a problem that k blocks $\{b_i | i = 1,2,...,k\}$ need to be H-aligned in a frame $W \times H$, where b_i has dimension $w_i \times h_i$ respectively. Let δ denote the dummy block and x_1 be the x coordinate of block b_1. The width of δ is

$$w_\delta = W - x_1 - \sum_{i=1}^{k} w_i$$

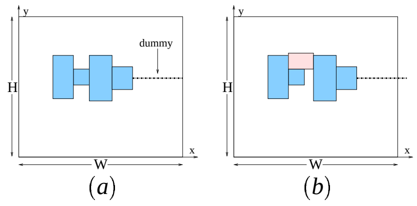

Figure 4: (a) Dummy block is introduced at the right by H-alignment constraint; (b) In violation of H-alignment constraint, the dummy block is pushed out of frame.

and the height of δ is $h_\delta = 0$.

If we handle multiple H-alignment constraints, another condition (called **non-crossing**) must be satisfied. For example, as shown in Fig 5, the two sets of blocks (a_i and b_i, $i = 1,2,3$) can not be H-aligned because a_i and b_i are crossing. However, both a_i and b_i satisfy the adjacency-order condition as stated in Theorem 1. Then we have the following necessary condition (non-crossing).

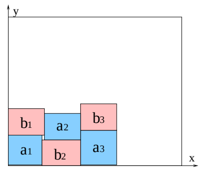

Figure 5: The two sets of blocks (a_i and b_i, $i = 1,2,3$) can not be H-aligned although they satisfy the adjacency-order condition in sequence pair representation. The sequence pair is $(X,Y) = (b_1\ a_1\ a_2\ b_2\ b_3\ a_3, a_1\ b_1\ b_2\ a_2\ a_3\ b_3)$.

THEOREM 2. **(non-crossing)** *Given any two sets of blocks, a_i, $i = 1,...,k$ and b_j, $j = 1,...,k'$, to be H-aligned, the corresponding sequence pair representation must satisfy that no relative position like $(...a_{i_1}...b_{j_1}...b_{j_2}...a_{i_2}...,...b_{j_1}...a_{i_1}...a_{i_2}...b_{j_2}...)$ appears in sequence pair (X,Y).*

Note that the two necessary conditions (Theorem 1 and Theorem 2) guarantee the topological relation of blocks for H-alignment constraints. The feasibility of constrained placement is actual-dimension dependent. However, the two conditions can be utilized to eliminate lots of infeasible sequence pairs.

For V-alignment constraint, we are dealing with the two sequences (X^R,Y) similarly. If the k blocks are V-aligned, the sequence pair must appear like $(X,Y) = (...b_k...b_{k-1}......b_i......b_1...,...b_1...b_2......b_i......b_k...)$ maintaining the reverse order in the first sequence and the obverse order in the second sequence. Analogously we have the following necessary conditions for V-alignment constraint in terms of sequence pair composition.

THEOREM 3. *If k blocks, b_i, $i = 1,2,...,k$, are V-aligned, then the corresponding sequence pair representation must satisfy that $(X^R,Y) = (...b_1...b_2......b_i......b_k...,...b_1...b_2......b_i......b_k...)$ and b_i is strictly ahead of b_{i+1} in (X^R,Y), $1 \le i \le k-1$.*

THEOREM 4. *Given any two sets of blocks, a_i, $i = 1,...,k$ and b_j, $j = 1,...,k'$, to be V-aligned, the corresponding sequence pair representation must satisfy that no relative position like $(...a_{i_1}...b_{j_1}...b_{j_2}...a_{i_2}...,...b_{j_1}...a_{i_1}...a_{i_2}...b_{j_2}...)$ appears in (X^R,Y).*

Note that "strictly ahead" for V-alignment is defined on sequences (X^R,Y). We have the following lemma.

LEMMA 1. *Given two sequences X_i and Y_j with positive weights, $lcs(X_i^R,Y_j) = 0$ implies $lcs(X_i,Y_j) = 0$.*

Let $(X,Y) = (X_0 b_k X_1 b_{k-1} X_2 ... b_i X_{k-i+1} ... b_1 X_k, Y_0 b_1 Y_1 b_2 Y_2 ... b_i Y_i ... b_k Y_k)$. Then $(X^R, Y) = (X_k^R b_1 X_{k-1}^R b_2 X_{k-2}^R ... b_i X_{k-i}^R ... b_k X_0^R, Y_0 b_1 Y_1 b_2 Y_2 ... b_i Y_i ... b_k Y_k)$. Therefore $lcs(X_{k-i}^R, Y_i) = 0, 1 \le i \le k-1$, which implies $lcs(X_{k-i}, Y_i) = 0, 1 \le i \le k-1$.

4. PERFORMANCE CONSTRAINT

Minimizing total wire length as traditional floorplanners did, can not guarantee bounded delay for critical nets. This becomes more important because timing convergence is a big issue in deep submicron design.

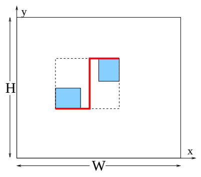

Figure 6: The bounding box of the farthest pin locations in a net.

Given a two-pin net with source at location (x_1, y_1) and sink at location (x_2, y_2), the timing requirement can be described as

$$t_1 + \lambda|x_2 - x_1| + \lambda|y_2 - y_1| \le t_2$$

where t_1 is the arrival time at source, t_2 denotes the required latest arrival time at sink, and λ represents the constant parameter for converting from Manhattan distance to timing domain. Thus we get

$$|x_2 - x_1| + |y_2 - y_1| \le (t_2 - t_1)/\lambda$$

Note that we use a linear function in terms of distance to estimate delay. This is reasonable[1]. For floorplanning where pin locations are not known, we assume the pin locations to be as far as possible, i.e., the source-sink distance equal to the half perimeter of the bounding box of the two blocks as shown in Fig 6. Note that this is a pessimistic estimation. An alternative way is to assume pin location at the center of a block or assigning a pin location on the closest boundary of a block, either of which can be handled in the same way.

For multi-pin net, we consider the longest distance between two nodes. In the paper, we use the half perimeter of the bounding box of the blocks belonging to the net to estimate the delay.

Based on the above formula, we can determine two values (how to determine will be discussed later), u and v, such that

$$\begin{aligned} & u + v = (t_2 - t_1)/\lambda \\ \text{and} \quad & |x_2 - x_1| \le u \\ \text{and} \quad & |y_2 - y_1| \le v \end{aligned}$$

Equivalently, the blocks belonging to the net are to be placed within a rectangular box (called range box) with width u and height v. It should be noted that the box is floating, and its shape can be changing with different ratio of u and v. Given a net connecting k blocks, b_i, $i = 1, 2, ..., k$, with coordinate (x_i, y_i) respectively, let $(\bar{x}, \bar{y})$ denote the coordinate of the box referring to the lower-left corner. Thus the coordinate of the box is

$$\begin{aligned} \bar{x} &\leftarrow min\{x_i | i = 1, 2, ..., k\} \\ \bar{y} &\leftarrow min\{y_i | i = 1, 2, ..., k\} \end{aligned}$$

Bounded delay may not be satisfied for some placement obtained by some sequence pair. If the delay of a net is out of bound, then the involved blocks are placed out of range box by the above measurement. In order to solve the problem, we add a dummy block at the right and another dummy block on top of all constrained blocks in a critical net, as shown in Fig 7(a). If one or more of the constrained blocks is placed out of range box, then either or both of the dummy blocks will be pushed out of frame as shown in Fig 7(b). Formally, the dummy blocks are determined as follows. Given a problem that a critical net connects k blocks $\{b_i | i = 1, 2, ..., k\}$ in a frame $W \times H$, a range box is determined at location $(\bar{x}, \bar{y})$ and with width u and height v where $(\bar{x}, \bar{y})$ is computed by above formula. Thus the right dummy block has the dimension:

[1]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.

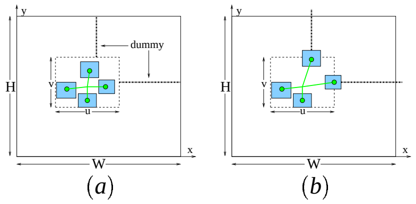

Figure 7: (a) Dummy blocks are introduced at the right or on top by performance constraint; (b) In violation of performance constraint, one or more of the dummy blocks is placed out of frame.

$$width = W - \bar{x} - u \qquad \text{and} \qquad height = 0$$

and the dimension of the above dummy block is:

$$width = 0 \qquad \text{and} \qquad height = H - \bar{y} - v$$

Now the remaining problem is how to determine u and v since the shape of the range box can change. We adjust the value of u and v dynamically in simulated annealing as follows. At high temperature in annealing process, we let u and v be half-half:

$$u = \frac{t_2 - t_1}{2\lambda} \qquad \text{and} \qquad v = \frac{t_2 - t_1}{2\lambda}$$

At low temperature, letting s be the smallest span in x axis that covers all of the constrained blocks $\{b_i | i = 1, 2, ..., k\}$, i.e. $s = max\{x_i + w_i | i = 1, 2, ..., k\} - min\{x_i | i = 1, 2, ..., k\}$, we adjust u and v:

$$u = min((t_2 - t_1)/\lambda, s) \qquad \text{and} \qquad v = (t_2 - t_1)/\lambda - u$$

This adjustment is certified by the following reasons. At high temperature simulated annealing is characterized as chaotic process where a square range-box is appropriate to use for approximate guidance. On the other hand, at low temperature a specific range box is used to exactly capture the meaning of delay bound.

5. ALGORITHM

5.1 Feasible Sequence Pair

Sequence pair always entails a packing if no constraints are concerned. However, when constraints are introduced, there may not exist corresponding packing for some sequence pairs. *Feasible sequence pair* can be defined as follows.

DEFINITION 4. **Feasible sequence pair**: *If there exists a packing which meets all the constraints and the topological relations imposed by a sequence pair, then the sequence pair is feasible. Otherwise, it is infeasible.*

5.2 Optimality

Although not all sequence pairs are feasible for constraints such as fixed-frame, alignment and performance constraint, we can still show that sequence pair does not miss any placement. Given a placement satisfying the constraints, a sequence pair can be obtained by gridding operation in [10]. This observation is stated in the following theorem.

THEOREM 5. *Any placement that satisfies the given constraints is represented by a sequence pair.*

Thus, an optimal packing with no violation of constraints is included in solution space of feasible sequence pairs. "Optimal packing" means that at least one area-minimal placement is represented by a sequence pair. This is not true in terms of wire length because packing blocks to the left and to the bottom may not represent the optimal wire length. However, packing is still a good method to minimize wire length because all blocks are packed, implying at least near-optimal wire length is represented.

5.3 Evaluation Algorithm

Note that the calculations of x and y coordinates of all the blocks are done independently by evaluating (X,Y) and (X^R,Y) respectively. In the following we mainly describe the algorithm to evaluate (X,Y) in the presence of constraints since the evaluation of (X^R,Y) can be done similarly.

The algorithm is based on the engine of computing longest common subsequence presented in [16]. We still compute longest common subsequence (lcs) in terms of the two sequence (X,Y) to determine the x coordinates of blocks. The alignment operation presented in Section 3 and the computation for performance-constrained blocks described in Section 4 can be embedded into lcs computation. Dummy blocks would not necessarily appear in sequence pair. Instead, a "out-of-frame" variable is introduced to record the intermediate *lcs* imposed by dummy blocks in constraints (alignment/performance). Note that the alignment operations for different alignment constraints are affected by each other. One iteration of lcs computation may not align all blocks imposed by alignment constraints. However, since there is no crossing between alignment constraints, one iteration of lcs computation will align at least one sequence of blocks related to an alignment constraint. Thus at most $k+1$ iterations are needed, where k is the total number of alignment constraints. On the other hand, if alignment constrained blocks are not aligned after $k+1$ iterations, then it indicates that there exists crossing between different alignment constraints. In typical applications, the number of alignment constraints, i.e. k, is bounded. Let $lcs'(X,Y)$ denote the return value of the algorithm, i.e. the length of the longest common subsequence with the presence of dummy blocks introduced by constraints. Note that we are dealing with fixed-frame floorplanning and thus it is enforced that $lcs'(X,Y) \geq W$, the width of the frame.

Similarly $lcs'(X^R,Y)$ is computed with dummy blocks in y direction, and $lcs'(X^R,Y) \geq H$, the height of the frame. If the total number of constrained blocks is $O(n)$ and the number of alignment constraints is bounded, the complexity of the algorithm is $O(n \log\log n)$, as stated in the following theorem.

THEOREM 6. *With the presence of fixed-frame/alignment/performance constraints, a sequence pair can be evaluated in $O(n\log\log n)$ time if the number of constrained blocks is $O(n)$ and the number of alignment constraints is bounded.*

In typical applications, the number of constrained blocks (e.g. the involved blocks in critical nets) is less than n. Note that a block may repeat in several critical nets. In general case, where there are k alignment constraints and m blocks (including repeated) are constrained, the time complexity of the algorithm will be $O(m + kn\log\log n)$. Note that even in the case where k is bounded, the method operating on constraint graph still needs $O(n^3)$ runtime.

5.4 A Necessary and Sufficient Condition

As presented in Section 3 and Section 4, the dimension of dummy blocks is determined by the frame for floorplanning. If there is any violation of given constraints (fixed-frame/alignment/performance), one or more of dummy blocks or real blocks will be pushed out of the frame. On the other hand, if all dummy blocks and real blocks are packed within the frame, then we can derive the corresponding sequence pair and compute a packing satisfying all constraints. Thus, we get the following necessary and sufficient condition.

THEOREM 7. *A sequence pair (X,Y) is feasible with constraints if and only if $lcs'(X,Y) \leq W$ and $lcs'(X^R,Y) \leq H$.*

Since we have enforced $lcs'(X,Y) \geq W$ and $lcs'(X^R,Y) \geq H$ in the evaluation, the following result is directly implied.

COROLLARY 1. *The necessary and sufficient condition of feasible sequence pair (X,Y) is that $lcs'(X,Y) = W$ and $lcs'(X^R,Y) = H$.*

5.5 Unified Cost Function

With all these constraints taken into account, the feasibility of a sequence pair depends on the actual dimension of blocks. Thus it is hard to judge the feasibility without evaluating a sequence pair. In other word, it is impossible to just go through solution space including feasible sequence pairs only. The necessary conditions for alignment constraint presented in Section 3 can be used to reduce the solution space although it can not eliminate all infeasible sequence pairs. An intuitive way to evaluate infeasible sequence pairs is to assign them an infinite cost. However, this will seriously affect the smoothness of stochastic search process. Note that if a sequence pair (X,Y) is infeasible then $lcs'(X,Y) > W$ or $lcs'(X^R,Y) > H$. Thus, we treat an infeasible sequence pair as if it is feasible, and regard the "area" as

$$A = lcs'(X,Y) \cdot lcs'(X^R,Y)$$

Therefore, we get a unified cost function

$$C = \alpha A + \beta W$$

where W is interconnect cost, and α and β are coefficients for balancing between A and W.

The unified cost function interprets the violation of infeasible sequence pair so naturally that we get the following benefits: (i) no need to accumulate the error for each constrained block; (ii) no need to add an additional penalty term into the cost function; and (iii) fixing an infeasible sequence pair (adaption) is not necessary. As a result, it helps handle constraints very fast and very well.

5.6 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. We maintain the relative ordering of alignment-constrained blocks in swapping. Swap can be done in linear time.

Rotation is to rotate an unconstrained block (e.g. exchange the width and height of the block). Rotation does not change the relative ordering imposed by alignment constraint. This operation can be done in constant time.

Flip is used to change the orientation of an alignment constraint. If a set of blocks is to be aligned horizontally, then flip the blocks to be aligned vertically, vice versa. The operation on sequence pair is to reverse the order of the set of blocks involved in the alignment constraint in the first sequence. In the implementation, we restrict that a block can only be associated with at most one alignment constraint. Thus, after flipping, a sequence pair still meets the ordering condition imposed by all alignment constraints. Suppose that an alignment constraint involves r blocks. Thus the flip operation can be done in $O(r)$ time. Of course, $r \leq n$.

In our implementation of the algorithm, the probability of the three operations is $P(\text{swap}) > P(\text{rotation}) > P(\text{flip})$.

5.7 Solution Space

The solution space we explore in simulated annealing includes only sequence pairs that satisfy the ordering condition imposed by alignment constraints. A set of move operation is said to be "complete" if any sequence pair in solution space can be reached by other sequence pair through a sequence of move operations. Then, the set of move operations proposed above is complete because a sequence pair can be transformed to another sequence pair via a sequence of swap and flip operations. However, rotation is also necessary for better packing.

Given any two different sequence pairs in solution space, one can be changed to the other as follows. First, we use flip operation to make the ordering of alignment-constrained blocks be the same in the two sequences. Then, one sequence can be "sorted" into the other via swap operations. Note that swapping two adjacent blocks does not change the relative ordering of alignment-constrained blocks unless the swap is invalid (i.e. the two adjacent blocks belong to the same alignment constraint). Since one alignment constraint involves at least 2 blocks, there are at most $n/2$ alignment constraints. Thus we need at most $n/2$ flip operations. Sorting requires at most $1+2+\ldots+(n-1) = n(n-1)/2$ adjacent swaps. In total, at most $n/2+n(n-1)/2 = n^2/2$ move operations are needed to transform a sequence pair to another in solution space. Therefore, the diameter of solution space is at most $n^2/2$ for the set of move operations.

5.8 Min-area Optimization

Traditional floorplanners use minimizing area as an objective. Although our algorithm is explicitly designed for fixed-frame floorplanning, it can also be adapted for min-area optimization. An initial frame, W and H under some given aspect ratio, is determined, e.g. $W \cdot H$ equals 120% of the sum of the areas of all the blocks. Then, we decrease W and H when a feasible sequence pair is met in simulated annealing. Three kinds of decreasing schemes are randomly chosen in our experiments: decrease W only, or H only or both.

6. EXPERIMENTAL RESULTS

Table 1: Min-area results of floorplanning with alignment/performance constraints.

circuit	block	constrained blocks		Results		
		align	perf	time (s)	area (mm^2)	dead space
apte	9	4	0	8	47.08	1.1%
xerox-1	10	4	0	9	20.16	4.0%
xerox-2	10	4	2	16	20.93	7.5%
hp-1	11	4	0	10	9.342	5.5%
hp-2	11	4	2	17	9.342	5.5%
ami33-1	33	4	0	31	1.221	5.3%
ami33-2	33	4	3	45	1.226	5.7%
ami49-1	49	5	0	41	38.20	7.2%
ami49-2	49	4	3	241	38.51	7.8%
ami49-3	49	4	6	278	38.72	8.4%

We have implemented the algorithm and tested on problems with various constraints. The experiments were carried out on a SUN Sparc Ultra 10(440Mhz). The technique of simulated annealing is used to search for an optimal placement with a special annealing schedule where a very large number of temperatures is used but only a small number of moves are made within each temperature.

The test problems are derived from MCNC benchmarks for block placement. We impose alignment/performance constraints on a set of blocks (determined by the total number of blocks in the benchmark). Table 1 lists the min-area results. Note that many other floorplanning algorithms give the results for area minimization. Our results are very comparable even though we have handled various alignment/performance constraints. There is no uniform standard for wire length measure because some assume pins at the center of blocks and some let pin locations on the boundary of blocks. Therefore, the results for wire length are omitted. Experimental results show that the algorithm performs very well in dealing with various alignment/performance constraints. The algorithm is very fast – we can easily evaluate a million sequence pairs with constraints in one minute for typical input size of placement problems. The experience in experiments tells us for min-area optimization it is preferable that the initial frame be close to the final one. As illustrations, Fig 8 displays the final packing results for ami49-3 .

7. CONCLUDING REMARKS

In this paper, we have presented an algorithm to handle alignment and performance constraints. The algorithm is based on sequence pair representation and evaluates a sequence pair with constraints in $O(n \log\log n)$ time typically where n is the number of blocks and the number of constrained blocks is $O(n)$. It is designed

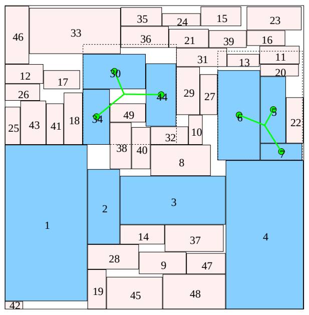

Figure 8: The result packing of ami49-3. Block 1, 2, 3 and 4 are to be aligned, and there are two performance constraints (block 5, 6 and 7 are connected in a critical net, and block 30, 34 and 44 are belonging to another critical net).

mainly for fixed-frame floorplanning optimization, and can also be adapted for minimizing area.

We have derived the necessary and sufficient condition of feasible sequence pair and a unified cost function for the evaluation of both feasible and infeasible sequence pairs. We also propose a set of move operations which is complete. Experimental results show that the algorithm performs very well in handling alignment and performance constraints.

Finally, we would like to point out that the algorithm to handle alignment constraint can apply to abutment constraint because abutment is a special case of alignment. The method proposed in the paper has been integrated in a system, FAST-SP, which previously handled constraints such as pre-placed, range, and boundary[16]. After integration, it is capable of handling more constraints (e.g., abutment, alignment, performance, etc.).

8. REFERENCES

[1] S.N. Adya and I.L. Markov. “Fixed-outline floorplanning through better local search”, ICCD-01, pp. 328-334, 2001.
[2] 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.
[3] P. Chen and E.S. Kuh. “Floorplan sizing by linear programming approximation”, DAC-00, pp. 468-471, 2000.
[4] K. Fujiyoshi, and H. Murata. “Arbitrary convex and concave rectilinear block packing using sequence pair”, ISPD-99, pp. 103-110, 1999.
[5] 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.
[6] 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.
[7] A.B. Kahng. “Classical floorplanning harmful”, ISPD-00, pp. 207-213, 2000.
[8] M. Kang and W.W.M. Dai. “Arbitrary rectilinear block packing based on sequence pair”, ICCAD-98, pp. 259-266, 1998.
[9] M. Lai and D.F. Wong. “Slicing tree is a complete floorplan representation”, DATE-01, pp. 228-232, 2001.
[10] 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.
[11] S. Nakatake, H. Murata, K. Fujiyoshi, and Y. Kajitani. “Module placement on BSG-structure and IC layout applications”, ICCAD-96, pp. 484-491, 1996.
[12] T. Ohtsuki, N. Sugiyama, and H. Kawanishi. “An optimization technique for integrated circuit layout design”, ICCST Kyoto, Japan, pp. 67-68, 1970.
[13] R.H.J.M. Otten. “Automatic floorplan design”, DAC-82, pp. 261-267, 1982.
[14] 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.
[15] 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.
[16] X. Tang and D.F. Wong. “FAST-SP: A fast algorithm for block placement based sequence pair”, ASPDAC-01, pp. 521-526, 2001.
[17] D.F. Wong and C.L. Liu. “A new algorithm for floorplan design”, DAC-86, pp. 101-107, 1986.