Timing Influenced Force Directed Floorplanning

Habib Youssef  Sadiq M. Sait  Khalid J. Al-Farra

Department of Computer Engineering
King Fahd University of Petroleum and Minerals
Dhahran-31261, Saudi Arabia
e-mail: youssef@ccse.kfupm.edu.sa

Abstract
We present a timing driven floorplanning program for general cell layouts. The approach used combines quality of force directed approach with that of constraint graph approach. A floorplan solution is produced in two steps. First a timing and connectivity driven topological arrangement is obtained using a force directed approach. In the second step, the topological arrangement is transformed into a legal floorplan. The objective of the second step is to minimize the overall area of the floorplan. The floorplanner is validated with circuits of sizes varying from 7 to 125 blocks.

1 Introduction
Floorplanning is an essential design step when a hierarchical/building block design methodology is used. Floorplanning helps solve problems such as overall required area, sizes and shapes of modules, pin and pad locations, etc. It is closely related to placement. Where for placement shape of modules and pin positions are fixed, in floorplanning these have some specified flexibility. The shape flexibility represents the designer's freedom to select one among several implementations of an element. The goal of floorplanning is to come up with a placement plan that will decide topological proximity as well as appropriate shapes and orientations of each block.

In most contemporary design methodologies, timing is of prime importance. Nowadays, it is rare that one finds a placement program which is not timing sensitive. Similar to placement, the floorplanning step has a dominant effect on circuit performance. It is of extreme importance to make floorplanning step timing-sensitive since this step helps decide several major questions with respect to the structure and timing performance of the circuit.

A possible approach to floorplan design is to first determine a topological arrangement of the blocks based on desired objectives (such as connectivity and timing performance), with no concern to geometric constraints. The second step is floorplan sizing, where sizes and shapes of the blocks are decided. The floorplanner described in this paper follows this approach. We obtain a floorplan in two steps: (1) construction of a timing driven topological arrangement using a force-directed approach; (2) conversion of the topological arrangement into a legal floorplan.

2 Problem Definition
The timing driven floorplanning problem can be formulated as follows:

Given:
1. A set of \( n \) rectangular blocks \( B = \{b_1, b_2, \ldots, b_n\} \). For each \( b_i \in B \) we have
   - \( w_i, h_i \): width and height of \( b_i \), which are constants for rigid blocks and variables for flexible blocks.
   - \( w_i^\text{min}, w_i^\text{max} \): lower and upper bounds on the width of \( b_i \) if \( b_i \) is a variable-shape block.
   - \( a_i \): area of \( b_i \) (i.e., \( a_i = w_i \times h_i \)), \( a_i \) is constant.
2. A set of nets \( N = \{n_1, n_2, \ldots, n_t, \ldots, n_k\} \) describing the connectivity information. Each \( n_i \) is assigned a bound \( u_i \) on its interconnect delay.
3. A set of timing critical paths, each having a certain slack.
4. Desirable floorplan aspect ratio: \( \rho = H/W \) where \( H \) and \( W \) are the height and width of the floorplan respectively.

Output:
A legal floorplan, that is, a floorplan satisfying the following constraints and objectives:
1. each block \( b_i \) is assigned to a location \( (x_i,y_i) \);
2. no two blocks overlap;
3. \( w_i^\text{min} \leq w_i \leq w_i^\text{max} \) and \( a_i = w_i \times h_i \) for each flexible block \( b_i \);
4. meet chip aspect ratio constraint;
5. meet timing constraints on the critical paths;
6. minimize chip area.
3 Literature Review

The intractability of the floorplanning problem has led to a large number of heuristic solution techniques. The traditional objectives include minimum chip area, minimum total wire length, routability, or a combination of these. Recently, circuit performance has become a popular objective.

In [1], a floorplan is obtained in two steps. The first step uses a sequence of gradient descent operations based on force-directed functions. The best floorplan is selected and simulated annealing is then applied in the second step to remove cell overlaps. Circuit timing is considered among other objectives of the floorplan. The procedure reported in [2] is constraint graph based and proceeds in two steps, where the dimensions of all the blocks are iteratively computed based on the length of the longest path in the constraint graph passing through the block. The floorplanner reported in [7] follows also a constraint graph approach. Constraint reduction and block reshaping are used to find floorplans with optimal areas. In [6], floorplanning is formulated as a linear mixed integer program with the objective of minimizing the overall area of the rectangle enclosing all the basic rectangles. In [3], a graph theoretic rectangular dualization approach to construct rectangular floorplans was presented. The authors transformed the rectangular dualization problem into a bipartite matching problem. Numerous other approaches have also been reported. The reader may refer to [5] for a detailed description of the many approaches used to address this problem.

4 Timing Predictions

A timing driven physical design tool expects necessary timing information. This information may consist of either or both of the following: (i) a list of the most critical paths, and (ii) timing constraints on all the nets. In this work, the timing analyzer produces both, a list of the most critical paths as well as a delay bound for each net. The critical paths are predicted according to the concept of α-criticality, which we will describe in this section. The timing bounds on nets are computed using the algorithm proposed in [8].

4.1 The α-Critical Approach

The total number of paths in a VLSI design grows exponentially with the size of the design. However, usually a small subset of these paths are timing critical. A path π is critical if its total delay, \( T_\pi \), is very close to its latest required arrival time \( LRA T_\pi \). If \( T_\pi \) exceeds \( LRA T_\pi \), path π becomes a long path. The path delay consists of two components: the logic delay which is known prior to layout, and the interconnect delay which is unknown. The interconnect capacitance is a key element in the total interconnect delay.

Let \( \pi = \{v_1, v_2, \ldots, v_p\} \) be a path in the circuit graph, where \( v_1 \) and \( v_p \) are the source and sink cells. The total delay on π is given by

\[
T_\pi = \sum_{i=1}^{p-1}(CD_{v_i} + ID_{v_i})
\]  

where, \( CD_{v_i} \) is the switching delay of cell \( v_i \) and \( ID_{v_i} \) is the interconnect delay of the net driven by cell \( v_i \), which may be expressed as follows,

\[
ID_{v_i} = LF_{v_i} \times C_{v_i}
\]  

where, \( LF_{v_i} \) is the load factor of the output pin of the driving cell \( t_i \), expressed in units of time per unit of capacitance, and \( C_{v_i} \) is the total interconnect capacitance (area + fringe) of the net driven by cell \( v_i \).

The interconnect capacitance \( C_{v_i} \), is estimated using data from past designs as follows. The average and standard deviation of net length for different types of nets (2-pin, 3-pin, ..., m-pin) were collected from past designs of similar complexity\(^1\). These are transformed into interconnect capacitances. Let \( C_{v_i} \) and \( s_{v_i} \) be the estimated expected interconnect capacitance and standard deviation of the net driven by cell \( v_i \). These are computed as follows,

\[
\bar{ID}_{v_i} = LF_{v_i} \times \overline{C_{v_i}}; \quad S_{v_i}^2 = LF_{v_i}^2 \times s_{v_i}^2
\]  

Under the assumption of statistical independence between the nets, the expected delay and variance on any path π can be expressed as follows,

\[
T_\pi = \sum_{i=1}^{p-1}(CD_{v_i} + \bar{ID}_{v_i}); \quad S_\pi^2 = \sum_{i=1}^{p-1}S_{v_i}^2
\]  

Let \( T_\text{max} \) be the expected delay of the longest path in the circuit, that is,

\[
T_\text{max} = \max_{\pi \in \Pi}(T_\pi)
\]  

where \( \Pi \) is the set of all paths in the circuit graph \( G \).

Definition 1 A path π is α-critical iff:

\[
T_\pi + \alpha S_\pi \geq T_\text{max}
\]

The parameter \( \alpha \) is supplied by the user and is interpreted as a confidence level. The higher \( \alpha \) is, the larger the number of reported paths will be, and the higher is the probability of capturing all the critical paths. Reasonable values of \( \alpha S_\pi \) are \( \leq 5 \) nano seconds.

4.2 Net Delay Constraint Computation

Layout tools work on individual nets as opposed to timing analysis tools which work on paths. Several algorithms have been proposed in the literature to transform the path constraints into constraints on nets. In this work, we used the Minmax algorithm proposed in [8] to transform the path timing constraints into upper bounds on net delays.

\(^1\)this classification helps reduce the sample variance around the mean
4.3 Topological Arrangement

The topological arrangement is obtained in a greedy fashion by adding one block at a time to the partial floorplan (point placement). The algorithm maintains three disjoint sets: a placed set, an adjacent set, and an unplaced set. The placed set contains the already positioned blocks. The set of blocks having common connections with the elements of the placed set constitute the adjacent set. The unplaced set contains the remaining blocks of the design. The algorithm starts by computing initial dimensions for the chip using the supplied aspect ratio and the total area of the blocks. The initial height and width of the chip are computed as follows,

\[ H_o = \rho \sqrt{\sum \alpha_i} \quad W_o = \frac{H}{\rho} \quad (7) \]

This is done to make the growth of the chip controlled by the supplied aspect ratio. Next, the algorithm selects a seed block. We experimented with three seed selections. The first seed selection was the block with maximum connections. The second seed selection was a block belonging to the most critical path. The third seed was a batch seed (a group of blocks belonging to the most critical path). The second and third seeds exhibited similar results, and produced superior floorplan solutions than those obtained with the first seed selection.

In any such greedy approach, one is faced with two main problems: (1) selection of a block for placement, (2) finding a suitable location for the block. The selection procedure combines two criteria: timing and connectivity. The timing criterion is an evaluation of the timing constraints on the nets connecting the block to the partial floorplan. The connectivity criterion is the number of interconnects between the block and the partial floorplan.

The delay bounds on the nets are transformed into timing costs. The timing cost is increasing with decreasing values of delay bound. The timing cost is defined as follows:

\[ \text{cost}_i = \frac{CLOCK - u_i}{CLOCK} \quad (8) \]

where, \(CLOCK\) is the clock period and \(u_i\) is the delay bound on net \(n_i \in N\). All the \(\text{cost}_i\)'s will be in the interval \([0,1]\).

A gain function that combines both timing and connectivity is evaluated for each unplaced block having connections to the partial placement. The gain function for block \(b_i\) is computed as follows:

\[ G_i = \sum_{j \in F_k} c_{ij} + \beta \sum_{i \in B_j, j \in N_k} (1 - p_j) \text{cost}_j \quad (9) \]

where,
- \(F_k\): set of blocks in the partial floorplan at step \(k\) of floorplanning (point placement).
- \(B_j\): set of blocks interconnected by net \(n_j\).
- \(N_k\): set of partial nets\(^2\) at step \(k\) of the floorplanning process;
- \(p_j\): percentage of placed blocks of \(B_j\), where \(0 \leq p_j \leq 1\). (\(p_j\) is initially zero and increases as more blocks of \(B_j\) get added to the partial floorplan);
- \(c_{ij}\): connectivity of block \(b_i\) to block \(b_j \in F_k\);
- \(\beta\): real positive weight coefficient.

The gain function \(G_i\) consists of two terms: connectivity of block \(b_i\) to the partial floorplan \(F_k\), and a weighted sum of timing costs on nets connecting \(b_i\) to blocks in \(F_k\). Selecting the block with the maximum number of connections will minimize the connection length on the floorplan. The weighted sum of timing costs (the second term in \(G_i\)) will favor the selection of those blocks that are on critical paths (i.e., high timing cost).

Unassigned blocks are selected one at a time. The block with maximum gain, \(G_i\), is selected next and passed to a Place_Block procedure for positioning. The Place_Block uses a force directed approach to determine the zero-force location of the selected block in the partial floorplan. Blocks are subjected to two attraction forces: a timing-based force and a connectivity-based force. The timing-based force is a function of the delay bounds on the nets interconnecting the blocks. The timing force \(f_i^t\) exerted by net \(i\) is set equal to its timing cost \(\text{cost}_i\). The smaller the timing bound, the higher is the attraction force. As a result, blocks that are connected by timing critical nets will be assigned locations in topological proximity. On the other hand, the connectivity-based force is directly proportional to the number of connections between the blocks. Thus, highly connected blocks will be placed close to each other. Let \(c_{ij}\) represent the number of connections between blocks \(b_i\) and \(b_j\). The exerted connectivity-based force on \(b_i\) due to \(b_j\) is given by,

\[ f_i^c = c_{ij} \quad (10) \]

Let \(b_i\) be the block selected at step \(k\) of the floorplanning process, and \(B_k = \{b_1, b_2, \ldots, b_{i-1}, b_i, \ldots, b_n\}\) be the set of blocks connected to \(b_i\) and which have already been positioned. Let \(f_i^t\) and \(f_i^c\) be the forces between \(b_i\) and \(b_j \in B_k\). The zero-force timing sensitive location, \((x_i^t, y_i^t)\), and the zero-force connectivity sensitive location, \((x_i^c, y_i^c)\), are computed as follows,

\[ x_i^t = \frac{\sum_{k=1}^{n} p_{k} f_{i}^{p_{k}} x_{k}}{\sum_{k=1}^{n} p_{k} f_{i}^{p_{k}}} \quad y_i^t = \frac{\sum_{k=1}^{n} p_{k} f_{i}^{p_{k}} y_{k}}{\sum_{k=1}^{n} p_{k} f_{i}^{p_{k}}} \quad (11) \]

\[ x_i^c = \frac{\sum_{k=1}^{n} p_{k} f_{i}^{c_{k}} x_{k}}{\sum_{k=1}^{n} p_{k} f_{i}^{c_{k}}} \quad y_i^c = \frac{\sum_{k=1}^{n} p_{k} f_{i}^{c_{k}} y_{k}}{\sum_{k=1}^{n} p_{k} f_{i}^{c_{k}}} \quad (12) \]

where, \(p_{k}\) is the percentage of placed blocks belonging to the net connecting \(b_i\) and \(b_k\). Finally, the target location for \(b_i\) is derived as follows,

\[ x_i = \alpha_1 x_i^t + \alpha_2 x_i^c \quad y_i = \alpha_1 y_i^t + \alpha_2 y_i^c \quad (13) \]

\(^2\) A net connects a set of blocks; if some of these blocks are already in the partial floorplan, we call the net a partial net.
where, $0 \leq \alpha_1, \alpha_2 \leq 1$, and $\alpha_1 + \alpha_2 = 1$; $(\alpha_1$ and $\alpha_2$ are weight coefficients to define the relative importance of $(x^*_1, y^*_1)$ and $(x^*_2, y^*_2)$).

In computing a block target location, the Place Block procedure considers the connections of the block to the I/O pads. I/O pads are assigned to sides of the floorplan, but without identifying their exact locations on the floorplan boundary. The I/O pads assigned to a particular side are assumed to reside in the middle of that side. The I/O pads of the circuit are distributed equally on the four sides. The side to which a particular pad is assigned is determined as follows. If the block is connected to some unassigned pad, we compute the target location without considering its connection to the unassigned I/O pad. Next, we assign this pad to the side that is closest to the computed target location. Finally, the block target location is recomputed taking into account the location of the I/O pad.

The force-directed approach is notorious for assigning several blocks to the same location. In case the target location of the new block is occupied, the new block is moved to the nearest free location.

The output from this Cluster Growth greedy algorithm is a list of the blocks with their $xy$-coordinates, and a list of the I/O pads with their assigned sides.

4.4 Floorplan Sizing

In this step, we are concerned with the actual floor-planning where block attributes (e.g., absolute locations for the variable shape blocks) are defined and constraints on geometric fit and aspect ratios are satisfied. The output from the Cluster Growth algorithm is a topological arrangement optimized for timing and connectivity. In order to get the final legal floorplan, we must remove all overlaps. This step is performed without undoing any of the decisions of the timing sensitive force-directed step, i.e., topological proximity between blocks in the solution produced by the first step is maintained.

We adopted a constraint-based approach to convert the topological arrangement into a legal floorplan. This floorplan sizing phase consists of two steps: (1) construction of constraint set, and (2) shape optimization.

Graph Construction

The topological arrangement is interpreted as a set of topological constraints. Two directed acyclic graphs are used to capture the constraint set: a horizontal constraint graph $G_H$ and a vertical constraint graph $G_V$. The vertex set of $G_H$ is the set of blocks plus two dummy vertices $T$ and $B$. Similarly, the vertex set of $G_V$ is the set of blocks plus two dummy vertices $T$ and $B$. The dummy vertices $T$, $R$, $T$, $B$ correspond, respectively, to the left, right, top, and bottom boundaries of the chip. The edge set of $G_H$ models the to-the-left/to-the-right relationships, while that of $G_V$ models the on-the-top/at-the-bottom relationships.

Two blocks are constrained if the center of one block must be to the left/below the center of the other. A constraint set is complete if there exists a directed path between every pair of blocks $(b_i, b_j)$ either in $G_H$ or in $G_V$. In a strong complete set each pair of blocks is adjacent either in $G_H$, in $G_V$, or both [7]. Two blocks are adjacent if they are connected by an edge. It is clear that a floorplan that satisfies a strongly complete set will have no overlaps. Obviously, for two blocks not to overlap, only one constraint (either in the horizontal or vertical direction) is necessary and sufficient. Two blocks are called overconstrained if they are constrained in both the horizontal and vertical directions. The existence of overconstrained blocks negatively affects the area optimality of the floorplan.

Our graph construction procedure is based on this key observation.

Definition 2 A constraint set $(G_H, G_V)$ is sufficient if there exists an edge between every pair of blocks $(b_i, b_j)$ either in $G_H$ or $G_V$.

Clearly, a sufficiently constrained set $(G_H, G_V)$ is a strongly complete set. The approach in [7] starts with an overconstrained set, i.e., a set with many overconstrained blocks. This set is then reduced to a sufficiently constrained set by removing redundant constraints from only the longest paths in $G_H$ and $G_V$. A problem with this approach is that the size of the overconstrained set could be very large, and hence may require large computing resources. We believe that a more efficient approach would be to build directly a sufficiently constrained set using a constructive (greedy) procedure. If two blocks are overconstrained (i.e., horizontally and vertically), the constructive procedure will constrain the two blocks in either the horizontal or the vertical direction. The selection is based on which of the constraints will lead to a smaller-area floorplan. This process generates a constraint set: (i.e., $G_H, G_V$) according to Definition 2 and, at the same time, eliminates all redundant constraints right from the beginning. This is in contrast to the algorithm in [7], where only redundant edges belonging to the longest paths in $G_H$ and $G_V$ are examined. Removing all redundant constraints produces a more compact floorplan.

The graph construction algorithm examines each pair of blocks and inserts topological constraints in $G_H$ or $G_V$ according to their centers. If two blocks $i$ and $j$ are constrained in the horizontal and vertical directions, an edge $(i, j)$ is inserted in each of $G_H$ and $G_V$. Next, the algorithm enumerates the longest path $P_H(i, j)$ ($P_V(i, j)$) that goes through the inserted edge $(i, j)$ in $G_H$ ($G_V$). The edge that yields the shorter path is retained and the other is removed. The other case is when the blocks $i$ and $j$ are constrained in only one direction (i.e., either to the left or below). In this case there is only one choice and therefore the algorithm inserts the edge in the corresponding graph.

The final step consists of resizing the variable shape blocks in order to optimize the floorplan area and satisfy the remaining constraints on block/chip aspect ratios. Resizing is conducted while maintaining the topological constraints stated in $G_H$ and $(G_V)$. 

159
Block Reshaping

The reshaping algorithm determines dimensions and positions of the blocks so that the floorplan area is minimized and constraints on block shapes are satisfied. This is achieved by iteratively resizing flexible blocks on the longest paths in the constraint graphs $G_H$ and $G_V$.

Let $\ell(\pi_H)$ and $\ell(\pi_V)$ be the lengths of the longest paths $\pi_H$ in $G_H$ and $\pi_V$ in $G_V$ respectively. Let block $b_i$ be such that $b_i \in \pi_V$ and $b_i \notin \pi_H$. If $\pi_H$ is the longest path traversing $b_i$ in $G_H$, then the width, $w_i$, of $b_i$ can be increased by an amount $\delta_{w}^{*} = \ell(\pi_H) - \ell(\pi_H')$ without increasing the overall area of the floorplan. $\delta_{w}^{*}$ is the maximum block width increment that is guaranteed not to cause an increase in the length of the longest path $\pi_H$ in $G_H$. But, since the block width has an upper bound $w_{i}^{\text{max}}$, then the legal $\delta_{w}^{*}$ is given by

\[ \delta_{w}^{\text{legal}} = \min(\delta_{w}^{*}, w_{i}^{\text{max}} - w_i) \quad (14) \]

Thus, the new dimensions $w_i', h_i'$ for block $b_i$ are derived as follows,

\[ w_i' = w_i + c \times \delta_{w}^{\text{legal}} \quad ; \quad h_i' = a_i/w_i' \quad (15) \]

where $c$ is a user specified positive real number ($c \leq 1$) to control how large the $x$-increment should be. After each $\delta_{w}^{*}$ ($\delta_{v}^{*}$) resizing step, the height (width) is reduced by $\delta_{v}^{*}$ ($\delta_{w}^{*}$) with no increase in the floorplan width (height). Resizing the blocks in small increments helps achieve a smaller floorplan with the correct aspect ratio. This will be at the expense of a slight increase in the runtime requirements of the program. In our experiments, we set this parameter to 0.5.

The resizing process is terminated if there are no more candidate blocks for reshaping.

After completing the resizing process, the horizontal and vertical graphs are traced to determine the final $xy$-locations of the blocks. The lower left corner of the floorplan is at the origin $(0,0)$. The lower left corner of block $b_i$ is placed at $(x_i, y_i)$ where $x_i$ is the longest $L$-to-$b_i$ path in $G_H$ and $y_i$ is the longest $B$-to-$b_i$ path in $G_V$.

Finally, the blocks are enclosed inside the smallest bounding rectangle with the desired aspect ratio $\rho$. The area of the bounding rectangle is the area of the floorplan.

5 Experiments and Discussion

The floorplanning approach described in this paper has been implemented in the C language. Experiments were run on a 33 MHz 80386 IBM PC. We experimented with several test cases (see Table 1). The smallest test case is a simple 7-block example. These test cases are implemented in standard cell design using a 2μ n-well SCMOS technology [4].

For the sake of experimentation, we treated the cells as general cells (i.e., flexible dimensions). The smallest test case is used to demonstrate the area quality of the solution produced by the floorplan sizing step. Figure 1(a) shows the floorplan as obtained from the timing-sensitive force directed step. We specified an overall aspect ratio $\rho = 1.00$. Figure 1(b) shows the final floorplan which has minimum area. Also, observe that the relative positions of the blocks in the force directed topological order (Figure 1(a)) are preserved in the final floorplan (Figure 1(b)).

For the computation of target locations in the Place_Block procedure, we set the parameters $a_1, a_2$ to 0.5. For the parameter $\beta$ in the gain function $G_i$, we experimented with the values 0, 1, and 4. Table 2 summarizes the results for all circuits. For example, for the highway circuit (traffic light controller with 45 blocks), when timing is included in the selection function (i.e. $\beta > 0$), the average connection length has decreased by 7% to 20% (row “net_len”). Also, the average remaining path slack (row “slack”) in Table 2 has improved by 40%. For this circuit, the timing analyzer reported 65 critical paths. The longest path has a slack of 2.179 nano seconds, while the 65th longest path had a slack of 13.358 nano seconds. All these paths were safe after floorplanning was guided by timing (row “paths%” in Table 2). On the other hand, even with timing driven floorplanning we could not get 100% satisfaction for net delay bounds (this is so for all test circuits). Achieving less than 100% net satisfaction does not mean that the design will definitely suffer from timing problems. The reason is that usually the loss on one net is accompanied by a gain on other nets. The total block area for this circuit is 88624. The floorplan area after resizing is shown in row “chip area” in Table 2. Figure 2 shows the floorplan of the traffic light controller. The total execution time for this example was 1:00 min.

We observed that, for all circuits, for moderate val-

<table>
<thead>
<tr>
<th>chip</th>
<th>Function</th>
<th>IOpads</th>
<th>Cells</th>
<th>CLOCK</th>
<th>C.Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adder</td>
<td>5</td>
<td>11</td>
<td>10 ns</td>
<td>13</td>
</tr>
<tr>
<td>Par1</td>
<td>Parity-16</td>
<td>17</td>
<td>20</td>
<td>48 ns</td>
<td>16</td>
</tr>
<tr>
<td>Par2</td>
<td>Parity-8</td>
<td>9</td>
<td>30</td>
<td>32 ns</td>
<td>100</td>
</tr>
<tr>
<td>High</td>
<td>TLC</td>
<td>11</td>
<td>45</td>
<td>20 ns</td>
<td>65</td>
</tr>
<tr>
<td>Frac</td>
<td>Multiplier</td>
<td>24</td>
<td>125</td>
<td>41 ns</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 1: Test cases statistics.

Figure 1: A 7-block example: (a) topological order; (b) legalized floorplan.
ues of the timing weight ($\beta$), the average net length (and thus area) as well as timing of the circuit improve. However, when $\beta$ is increased too much, it is counter-productive for both average net length and circuit timing. The reason is that, increasing values of $\beta$ beyond the circuit requirement causes most nets to become excessively long, and hence the circuit timing gets worse.

One main advantage of our approach is that it is not restricted to slicing structures. Moreover, the execution time of the algorithm is very small. Our system was able to generate a legal floorplan for a 125-block circuit in less than 3 minutes on a 33 MHz 80386 IBM PC. This makes the algorithm a good tool for generating good initial solutions for iterative improvement algorithms (such as genetic algorithm, simulated annealing, etc).

6 Conclusion

In this paper we described a timing driven floorplanning program. Two types of timing data are used: a set of the $\alpha$-critical paths and delay upper bounds on the interconnects. A floorplan solution is constructed in two phases. The first phase builds a timing influenced topological arrangement of the blocks using a constructive implementation of the force directed approach. The second phase transforms this topological arrangement into a legal floorplan. The overall objective of the program is to produce a floorplan solution optimized for area, wirelength, and timing performance. Experiments on test circuits produced consistently optimized floorplans with respect to the aforementioned objective.

Acknowledgments

Authors acknowledge King Fahd University of Petroleum and Minerals for support under KFUPM Project # COE/VLSIDESIGN/162.

References


