A Performance Driven Macro–Cell Placement Algorithm†

Tong Gao, P. M. Vaidya, C. L. Liu

Department of Computer Science
University of Illinois at Urbana–Champaign

Abstract

In this paper, we present a new performance driven macro–cell placement algorithm. Placement of modules is guided by a set of upper– and lower–bounds on the net wire lengths. A convex programming algorithm is used to compute a set of upper–bounds on the net wire lengths which will ensure that timing requirements between input and output signals are satisfied. A set of lower–bounds is also computed to control signal skews at intermediate points of the circuit. Artificial nets are introduced between all pairs of modules. Lower–bounds on the lengths of the artificial nets are computed to avoid module overlaps in the placement. A modified min–cut placement algorithm is then used to generate a placement that satisfies the upper– and lower–bounds. We then employ an iterative procedure to modify the set of upper– and lower–bounds to improve the quality of the placement result.

1. Introduction

As technology advances, the size of modules in integrated circuits becomes smaller, while more modules are included in the circuits. Consequently, while the effect of intra–module delay becomes less significant, the effect of inter–module wire delay becomes more prominent. Thus, minimizing interconnecting wire delay becomes an important issue in improving VLSI circuit performance. Also, for a module to operate properly, input signals to the module must arrive more or less at the same time. The time difference between the latest arriving signal and the earliest arriving signal at a module is called the ***signal skew*** at the module. Both wire delays and signal skews depend on net wire lengths which in turn depend heavily on the physical placement of the modules. It is, therefore, important to study ***performance driven placement algorithms*** which can produce placements that satisfy the timing requirements of a design. Performance driven placement problems are in general difficult because of the incompatibility between timing requirements and physical requirements. Timing requirements are path oriented since delay between two points in the circuit depends on all the paths between them, while physical placement is net oriented since net wire lengths are determined by the placement of the modules.

The performance driven placement problem has been studied extensively in the past [2–13]. Most previous work can be grouped into two major categories, namely, net–based approach and path–based approach. In a path–based algorithm [3,4,5], timing requirements are considered explicitly and the algorithm tries to satisfy both timing requirements and physical requirements simultaneously during placement. In a net–based algorithm [6–11], timing requirements are first translated into physical requirements, such as net constraints. The placement algorithm then tries to generate a placement that satisfies the timing requirements under the guidance of the net constraints. One typical net–based approach is to first translate timing requirements into a set of upper–bounds on net wire lengths [6,9,10]. The upper–bounds are then used to guide the placement process. Utilization of upper–bounds solves the incompatibility problem between timing requirements and physical requirements. However, since timing requirements are path oriented, representing timing requirements by net constraints usually over–constrains the placement problem which in turn, might exclude some feasible placement solutions. In [11], an iterative approach was introduced to find a feasible placement solution that might have been excluded by a particular choice of upper–bounds. However, the algorithm in [11] considers only timing delays between input and output signals. Also, it only deals with gate–array technology.

In this paper, we present a new macro–cell placement algorithm which besides considering timing delay between input and output pins, also includes signal skew as part of the timing requirements. In our algorithm, upper–bounds on wire lengths of signal nets are computed to ensure that timing delay requirements are satisfied, lower–bounds on wire lengths of signal nets are computed to ensure that signal skew requirements at the modules are satisfied. Furtheremore, artificial nets are introduced between all pairs of modules. Lower–bounds on the wire lengths of these nets based on the dimensions of the macro–cells are introduced to control module overlaps during placement. A new modified min–cut algorithm is then used to generate placements that satisfy the upper– and lower–bounds on the net wire lengths. By introducing bounds on net wire lengths, we are able to incorporate all the timing requirements and physical requirements in an unique way. In our algorithm, we also employ an iterative approach to modify the upper– and lower–bounds according to the previously generated placement in order to improve the quality of the placement result.

2. Problem formulation

Input to the placement algorithm consists of circuit specification and timing specification. A circuit consists of a set of m modules $\{M_1, M_2, \cdots, M_m\}$, a set of n nets $N = \{N_1, N_2, ..., N_n\}$, and a set of p primary input pins (PIs) and primary output pins (POs) $\{R_1, R_2, ..., R_p\}$. Module M_i has dimensions $w_i \times h_i$, and is to be placed anywhere on the chip as long as it does not overlap with another module. The placement area has dimensions $W \times H$. A net is an interconnection of a set of pins on the boundaries of the modules, and net wire length is estimated by the half perimeter of the bounding rectangle of the pins in the net. For simplicity in computation, it is assumed that all pins are located at the centers of the modules and a net is viewed as an interconnection of the modules where the pins reside. Both primary input pins and primary out-

† This work was partially supported by the National Science Foundation under grants MIP 89–06932, CCR–9057481, and CCR–9007195.

put pins have predetermined positions on the boundary of the placement area. The jth pin of module M_i is denoted P_{ij}. Timing specification of the circuit includes signal arrival times at the PIs (the signal arrival time at PI R_i is denoted a_i), the required signal arrival times at the POs (the required signal arrival time at PO R_k is denoted r_k), and the maximum allowable signal skew C_i at module M_i. Internal delays of the modules are given constants. We use d_i to denote the internal delay of module M_i. Delay in a net is proportional to the lumped capacitance of the net, which in turn is proportional to the net wire length. To simplify our notation, we consider net wire length and net delay as equivalent throughout this paper. The net wire length of net N_i is denoted z_i, and the signal skew at module M_i is denoted c_i.

The timing problem has been studied in [1]. For a combinational circuit without feedback loop, the arrival time a_{ij} of the signal at pin P_{ij} and the arrival time a_k at primary output pin R_k is computed by a forward traversal of the circuit starting from the PIs. It is assumed that the early arriving signals wait for the latest arriving signal at the modules. The signal skew c_i at module M_i can be computed by taking the time difference between the latest arriving signal and the earliest arriving signal at module M_i. Obviously, a given placement satisfies the timing requirements if $a_k \leq r_k$ for all POs R_k and $c_i \leq C_i$ for all modules M_i.

It was shown in [11] that the timing delay requirements can be expressed as a set of linear constraints

$$a_i + \sum_{all\ M_u} d_u + \sum_{all\ N_v} z_v \leq r_k \qquad (2.1)$$

for all possible paths from any primary input pin R_i to any primary output pin R_k where the M_u's and N_v's are the modules and nets on the path from R_i to R_k, respectively.

If module M_i has only one input signal, the signal skew c_i is equal to 0. If M_i has $q > 1$ input pins $\{P_{i1}, P_{i2}, \ldots, P_{iq}\}$, the signal skew requirement $c_i \leq C_i$ can be expressed as a set of linear constraints

$$a_{ij} - a_{il} \leq C_i \qquad (2.2)$$

for all input pins P_{ij} and P_{il}, $j \neq l$, which in turn, are equivalent to

$$(a_s + \sum_{all\ M_u} d_u + \sum_{all\ N_v} z_v) - (a_t + \sum_{all\ M_w} d_w + \sum_{all\ N_y} z_y) \leq C_i \qquad (2.3)$$

for all the possible paths from any primary input pin R_s to pin P_{ij} and all possible paths from any primary input pin R_t to pin P_{il}. M_u's and N_v's are the modules and nets on the path from R_s to P_{ij}, respectively. M_w's and N_y's are the modules and nets on the path from R_t to P_{il}, respectively.

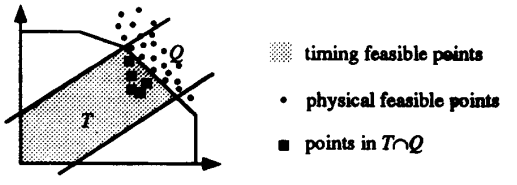

Figure 1. Relation between T and Q.

It is, therefore, possible to express the timing requirements as a set of linear constraints on the net wire lengths. A set of *positive* wire lengths $\hat{X} = \{\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_n\}$ for the nets in $N = \{N_1, N_2, \ldots, N_n\}$ is said to be *timing feasible* if it satisfies the linear constraints in (2.1) and (2.3), and is said to be *physical feasible* if there is a physical placement with net wire lengths equal to that in $\hat{X}$. A set of wire lengths $\hat{X}$ can be represented by a point in n dimensional space. Let T denote the set of all timing feasible points which will be referred to as the *timing feasible region* (Figure 1), and Q denote the set of all physical feasible points which will be referred to as the *physical feasible region* (Figure 1). A solution to the placement problem should have its net wire lengths corresponding to some point in $T \cap Q$ (Figure 1). Note that T is a convex polytope bounded by linear constraints whereas Q can be either a continuous region or a set of discrete points depending on the physical constraints in the placement problem.

3. Computing the bounds

Let $\hat{X} = \{\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_n\}$ be a timing feasible point. Consider the box bounded by the following hyperplanes which we shall denote $B_{\hat{X}}$:

$$z_1 = \hat{z}_1,\ z_2 = \hat{z}_2,\ \ldots,\ z_n = \hat{z}_n$$

$$z_1 = 0,\ z_2 = 0,\ \ldots,\ z_n = 0.$$

Since the linear constraints in (2.1) have only positive coefficients, any point in $B_{\hat{X}}$ satisfies the constraints in (2.1) (Figure 2). On the other hand, since there are negative coefficients in (2.3), some of the points in $B_{\hat{X}}$ might not satisfy the constraints in (2.3) (e.g., point Y in Figure 2). Therefore, if $\hat{X}$ is used as a set of upper-bounds, a physical placement with net wire lengths satisfying the upper-bounds is guaranteed to satisfy the timing delay requirements, but is not guaranteed to satisfy the signal skew requirements. To enforce the signal skew requirements, we also compute a set of lower-bounds on the net wire length. Initially, the lower-bounds for all signal nets are set to 0. As will be discussed in section 6, they will be modified iteratively to control the signal skews in the circuit.

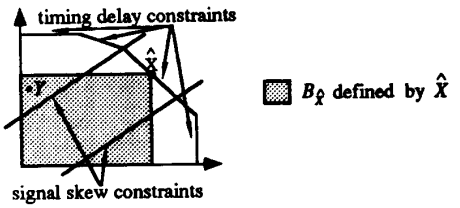

Figure 2. Upper-bounds and timing constraints.

As was suggested in [11], among the many possible choices of a timing feasible point $\hat{X}$ as a set of upper-bounds, we choose the one that maximizes the volume of $B_{\hat{X}}$ which shall allow maximum flexibility in physical placement. Consequently, we want to choose a point that is an optimal solution to the problem of maximizing the product $z_1 z_2 \cdots z_n$ subject to the linear constraints in (2.1) and (2.3). We use, instead, the objective function

$$\alpha_1 \log z_1 + \alpha_2 \log z_2 + \cdots + \alpha_n \log z_n$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are constants determined by the algorithm and are modified iteratively during successive iterations. The constants $\alpha_1, \alpha_2, \ldots, \alpha_n$ give us the flexibility to choose different sets of upper-bounds for different values of these constants. When all the α_i's are set to 1, maximizing the objective function

$$\alpha_1 \log z_1 + \alpha_2 \log z_2 + \cdots + \alpha_n \log z_n$$

is equivalent to maximizing the function $z_1 z_2 \cdots z_n$.

To solve the convex programming problem, we used a powerful convex programming algorithm described in [14]. The details of the algorithm can be found in [11,14], here we shall give only a brief summary of the algorithm.

The convex programming algorithm starts with an initial polytope β which encloses the optimal solution (Figure 3a), and iteratively generates cutting planes which cut off parts of β that do not contain the optimal solution (Figures 3b, 3c). In each iteration, a cutting plane is generated as follows: First, the center (see [14] for definition) of the polytope β is computed and its feasibility is checked by running a timing analysis for the circuit using the coordinates of the point as net wire lengths. If the center is feasible (e.g., all the timing requirements are satisfied), a cutting plane tangent to the surface of the objective function at the center is generated (Figure 3b). If the center is not feasible, a violated constraint in (2.1) or (2.3) is found and a cutting plane passing through the center and parallel to the constraint is generated (Figure 3c). The cutting plane divides the polytope β into two parts, and the cutting process continues for the part of the polytope that contains the optimal solution until the polytope β is small enough that a good estimation of the optimal solution can be made.

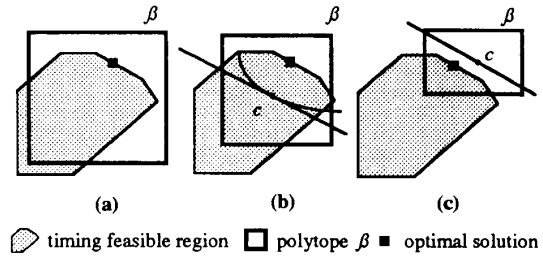

Figure 3. Computing a set of upper-bounds.

4. Minimizing overlaps between macro-cells

To control module overlaps during placement, we specify a minimum separation between each pair of modules. The minimum separation is expressed by the Manhattan distance between the centers of the two modules. Because the placement algorithm deals only with net wire lengths rather than distances between modules, an artificial net is introduced between each pair of modules. Denote the kth artificial net by N_{n+k}. (Nets N_1, N_2, ..., N_n are signal nets in the circuit.) The minimum separation requirement between a pair of modules is expressed as a lower-bound on the length of the corresponding artificial net. Note that there is no upper-bound on the length of an artificial net.

To guarantee that M_i and M_j will not overlap, it is sufficient to have $x_{n+k} \geq (w_i + w_j + h_i + h_j)/2$ where x_{n+k} is the wire length of the artificial net between M_i and M_j. Clearly, excessively stringent requirements might cause the placement algorithm to fail even though a feasible placement does exist.

To avoid over-constraining the placement problem, five different lower-bounds are computed for each artificial net according to the sizes and shapes of the modules the net connects. The five lower-bounds for net N_{n+k} are:

$$l_{k1} = (w_i + w_j)/2, \quad l_{k2} = (h_i + h_j)/2$$

$$l_{k3} = (w_i + h_j)/2, \quad l_{k4} = (h_i + w_j)/2$$

$$l_{k5} = (w_i + w_j + h_i + h_j)/2.$$

Note that each of the first four lower-bounds corresponds to a non-overlapping arrangement of M_i and M_j. In the placement phase, the placement algorithm tries to find a placement that satisfies as many of the lower-bounds as possible for each artificial net. The reason to have the lower-bounds l_{k3} and l_{k4} is that the placement algorithm allows a module to be rotated by 90 degrees. Obviously, the more lower-bounds that are satisfied, the better the chance that the modules will not overlap. Note that for the artificial nets it is not necessary to satisfy all the lower-bounds in order to have a non-overlapping placement.

5. The placement algorithm

After all the upper- and lower-bounds are computed, a placement algorithm is used to generate a placement that satisfies these bounds. A modified min-cut algorithm which tries to satisfy as many bounds as possible is used. In the placement process, the macro-cells are treated as points. The lower-bounds on net wire lengths of the artificial nets provide an effective way to handle variable sized modules as well as the case of non-gridded placement.

Basically, our placement algorithm is a modified min-cut algorithm. At each level of the partitioning hierarchy, net wire lengths are estimated and the estimated values are compared with their corresponding upper- and lower-bounds. Based on the result of the comparison, different weights are assigned to different nets. The weights are then used to compute the gains of the modules. Here we define the gain of a module to be the sum of the weights of the nets that are removed from the cut-set minus the sum of the weights of the nets that are added to the cut-set when the module is moved to its complementary partition.

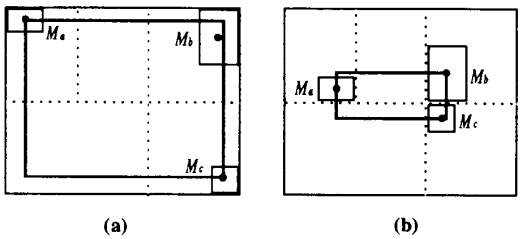

Figure 4. $\max_i$ and $\min_i$ for N_i connecting M_a, M_b, and M_c.

At any intermediate level of the partitioning hierarchy, the maximum and minimum possible wire lengths for each net are computed. Because a module can eventually be placed anywhere within its partition in the chip, the maximum wire length $\max_i$ for net N_i is the half perimeter of the smallest bounding rectangle that contains all the points at which the centers of the modules in the net may locate (Figure 4a), and the minimum possible wire length $\min_i$ is estimated as the half perimeter of the smallest bounding rectangle that contains at least one possible location for the center of each module in the net (Figure 4b).

The weight f_i for signal net N_i is the sum of f_{i1} and f_{i2} where f_{i1} and f_{i2} are computed as follows: Recall that $\hat{x}_i$ denotes the upper-bound on the net wire length of N_i. If $\max_i \leq \hat{x}_i$, f_{i1} is set to 0 because no matter how the partitions are further refined, the upper-bound $\hat{x}_i$ will be satisfied. If $\min_i > \hat{x}_i$, f_{i1} is set to a predetermined constant *mazcrit1* because no matter how the partitions are further refined, the wire length will exceed $\hat{x}_i$. If $\max_i \geq \hat{x}_i \geq \min_i$, f_{i1} is computed according to the formula

$$round\left(\frac{\max_i - \hat{z}_i}{\max_i - \min_i} \times maxcrit1\right)^{\dagger}.$$

Similarly, for the lower-bound l_i on the net wire length of N_i, if $\min_i \geq l_i$, then f_{i2} is set to 0. If $\max_i \leq l_i$, then f_{i2} is set to $-maxcrit1$. If $\min_i \leq l_i \leq max_i$, f_{i2} is computed according to the formula

$$-round\left(\frac{l_i - \min_i}{\max_i - \min_i} \times maxcrit1\right).$$

Next, we discuss how the weight f_k is computed for artificial net N_{n+k}. Because the artificial nets are introduced to ensure minimum separation between modules, it is desirable to have them included in the cut-set. Thus, the weight for an artificial net should always be non-positive. The value of f_k lies within a predetermined range, from $-maxcrit2$ to 0. It is computed as follows: If the two modules M_i and M_j connected by N_{n+k} belong to different partitions, f_k is set to 0 because these two modules will not overlap. If these two modules are in the same partition of dimensions $width \times length$, f_k is computed as

$$round\left(-maxcrit2 \times \frac{9-u}{9}\right)$$

where u is, among the 9 inequalities listed below, the number of inequalities that are satisfied.

$$width - w_i/2 - w_j/2 \geq l_{k1}, \quad length - w_i/2 - w_j/2 \geq l_{k1}$$

$$width - h_i/2 - h_j/2 \geq l_{k2}, \quad length - h_i/2 - h_j/2 \geq l_{k2}$$

$$width - w_i/2 - h_j/2 \geq l_{k3}, \quad length - w_i/2 - h_j/2 \geq l_{k3}$$

$$width - h_i/2 - w_j/2 \geq l_{k4}, \quad length - h_i/2 - w_j/2 \geq l_{k4}$$

$$length + width - w_i/2 - h_i/2 - w_j/2 - h_j/2 \geq l_{k5}$$

Because a module can be rotated by 90 degrees, each of the inequalities corresponds to a possible non-overlapping placement of the two modules. Thus, if more of these inequalities are satisfied, the algorithm will have a better chance of finding a non-overlapping placement of the two modules.

After the gains are computed, Fiduccia's min-cut algorithms is called to divide each partition, A, into two sub-partitions, B and C. Since we are partitioning a collection of macro-cells, some area balancing rule is needed. Experimental results suggested that

$$area_A/2 - maxmod \leq area_B \leq area_A/2 + maxmod$$

is a reasonable balancing rule where $maxmod$ is the area of the largest module in partition A. After A is partitioned into B and C, the cut line is shifted to further balance the areas of the partitions according to the areas of the module in B and C. Suppose that the coordinates of the lower left corner of partition A are (llx, lly) and the coordinates of the upper right corner of partition A are (urx, ury). If the cut line is vertical, the new cut line will be positioned at

$$llx + \frac{area_B(urx - llx)}{area_B + area_C}.$$

If the cut line is horizontal, the new cut line will be positioned at

$$lly + \frac{area_B(ury - lly)}{area_B + area_C}.$$

Note if the shapes of the modules are flexible, by shifting the cut line, it is guaranteed that each partition will have enough area for all the modules in the partition.

The partitioning process continues on the sub-partitions until each sub-partition contains at most one module and the area of the sub-partition is small enough so that a good estimation of the module position can be made.

Even though the modified min-cut placement algorithm tries to satisfy as many of the upper- and lower-bounds as possible, it does not guarantee that all of them will be satisfied. Therefore, if the shape of the macro-cells are rigid, there might be some overlaps in the placement generated. These overlaps can be removed by local movement and rotation of the modules. Because the modules are moved locally, the timing performance of the placement will not be changed substantially.

6. Iterative modification of the bounds

Even though the placement algorithm tries to satisfy as many of the upper- and lower-bounds as possible, it does not guarantee that all the timing requirements will be satisfied. Therefore, after a placement is generated, a timing analysis is carried out for the placement obtained. If the placement satisfies all the timing requirements, then we are done. Otherwise, a violated timing requirement is found. Depending on whether it is a timing delay requirement or a signal skew requirement, either the set of upper-bounds or the set of lower-bounds will be modified for the next iteration of the placement algorithm. By adjusting the bounds in accordance with information obtained from previous placements, we are able to improve the quality of placement iteratively.

As was pointed out in [11], physical feasible points are clustered in groups. If the placement algorithm fails to generate a physical placement that satisfies the timing delay requirements, the set of upper-bounds should be moved closer to the physical feasible point corresponding to the physical placement generated. This is achieved in our algorithm by adjusting the weights α_i's in the objective function $\alpha_1 \log x_1 + \alpha_2 \log x_2 + \cdots + \alpha_n \log x_n$. The new α_i's are computed as

$$\alpha_i = \alpha_i' \times (1 - (\hat{z}_i - \hat{y}_i)/q)$$

where α_i' is the old weight for x_i in the objective function, $\hat{y}_i$ is the wire length of net N_i in the placement, $\hat{z}_i$ is the old upper-bound for net N_i, and $q = \max_{i=1 \text{ to } n} |\hat{z}_i - \hat{y}_i|$. Since in the new objective function the coefficients are larger for nets the wire lengths of which exceeded the corresponding upper-bounds, the convex programming algorithm will try to increase the values of the upper-bounds for these nets.

On the other hand, if a signal skew requirement is violated (Figure 5a,b), in order to force the point corresponding to the physical placement (e.g., Y in Figure 5a,b) into the timing feasible region, we can either decrease the value of the upper-bounds as shown in Figure 5a (changing the upper-bounds from $\hat{X}$ to $\hat{X}'$) or increase the value of the lower-bounds as shown in Figure 5b. If we decrease the value of the upper-bounds on wire lengths for some nets, the set of upper-bounds for all the nets need to be recomputed (Figure 5a). Therefore, we choose to increase the values of the lower-bounds in the case that a signal skew requirement is violated. If the signal skew requirement at M_i is violated, by backtracking from the net with the earliest arriving signal at M_i, we can find a path p from some primary input pin R_j to M_i the length of which

---

† $round(x)$ represents the smallest integer greater than x if the fractional part of x is greater than 0.5, and represents the largest integer less than or equal to x otherwise.

should be increased in order to satisfy the constraint $c_i \leq C_i$. Consequently, the lower-bounds for the nets on path p should be increased so that

$$a_j + \sum_{all\ N_k \in p} l_k + \sum_{all\ M_k \in p} d_k = a_{il} - C_i$$

where a_j is the signal arrival time at R_j and a_{il} is the arrival time of the latest arriving signal at M_i. In order to give the placement algorithm maximum flexibility during placement, the quantity $q = a_{il} - C_i - a_j - \sum_{all\ M_k \in p} d_k$ should be distributed over the lower-bounds of the nets on p such that $\prod_{all\ N_a \in p} (\hat{z}_a - l_a)$ is maximized. It can be shown that the solution to this optimization problem is to set the value of the lower-bounds for each net N_a on p to be

$$l_a = \frac{q\ \hat{z}_a}{\sum_{all\ N_k \in p} \hat{z}_k}.$$

We omit the proof of this result here.

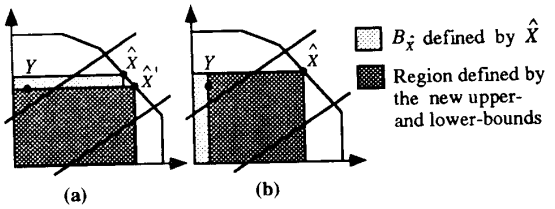

Figure 5. Modifying the upper- or lower-bounds.

7. Experimental result

Our algorithm was implemented in C and executed on a Sparc 2 workstation. We present the results on eight test examples the specification of which are summarized in Table 1. The columns "req." and "skew" are the required arrival time at the POs and the maximum allowable signal skew at the modules, respectively. Although our program is capable of handling different arrival times and required times at the PIs and POs, for simplicity of presentation of our results, we set the arrival times at all PIs to 0 and required time at all POs to be the same in our test runs. Also, even though our program is capable of handling different maximum allowable signal skew constraints at different modules, we set them to the same value.

circuit no.	no. of modules	no. of nets	W	H	req.	skew
1	72	145	16	16	100	20
2	83	172	16	16	165	20
3	97	197	16	16	110	22
4	97	222	16	16	110	19
5	102	213	16	16	110	22
6	102	236	16	16	110	22
7	104	219	16	16	115	22
8	102	229	16	16	115	22

Table 1. Circuit specifications.

Table 2 lists the results of the test runs. The result produced by Fiduccia's min-cut algorithm [15] is also listed for comparison. In Table 2, the column "arrive" is the latest arrival time at the primary outputs, the column "skew" is the maximum signal skew at all the modules in the circuit, and the column "wire" is the total wire length of the placement. On the average, our algorithms has a 26.0 percents improvement on input-output delays and a 42.0 percents improvement on the maximum signal skews when compared with the min-cut algorithm. Note that there is usually a trade off between timing performance and total wire length of the placement. (Since we are concerned with both timing delays and signal skews, improving timing performance does not always reduce total wire length.)

circuit no	min-cut algorithm			our algorithm		
	arrive	skew	wire	arrive	skew	wire
1	88.1	26.6	910.5	68.6	19.5	882.6
2	171.5	48.8	1057.3	107.9	19.6	1079.3
3	105.2	34.4	917.3	85.3	21.0	1043.2
4	136.1	42.3	1045.2	84.9	16.7	1117.6
5	96.2	31.1	1043.8	85.0	21.9	1242.5
6	132.7	47.9	1334.9	86.6	20.5	1333.6
7	94.3	25.9	1057.2	83.2	20.6	1235.5
8	116.2	39.3	1165.4	78.0	19.8	1213.6

Table 2. Results.

To illustrate the iterative procedure presented in section 6, intermediate results of each iteration when circuit 8 was placed are listed in Table 3. The row "spec." is the specified required time at the primary output pins and maximum allowable signal skew of the circuit. After the first iteration, both the timing delay and the signal skew requirements are not satisfied. Therefore, the set of upper-bounds was modified. After the second iteration, the timing delay requirements are satisfied but the signal skew requirements are still not satisfied. Therefore, the set of lower-bounds was modified. In the third iteration, the placement generated satisfies all the timing requirements. The placement generated by our algorithm is shown in Figure 6.

circuit 8	arrive	skew
spec.	115.0	22.0
iter. 1	120.6	44.2
iter. 2	82.5	31.2
iter. 3	78.0	19.8

Table 3. Intermediate results of placing circuit 8.

The average running time for computing the upper-bounds and generating a placement for each circuit are listed in Table 4. Note that the computation of the upper-bounds consumes a large protion of the total computation time.

circuit no.	1	2	3	4	5	6	7	8
upper-bounds	3.5	4.9	6.2	7.5	7.2	8.5	7.5	8.0
placement	0.3	0.4	0.5	0.4	0.5	0.6	0.5	0.5

Table 4. Running time in minutes.

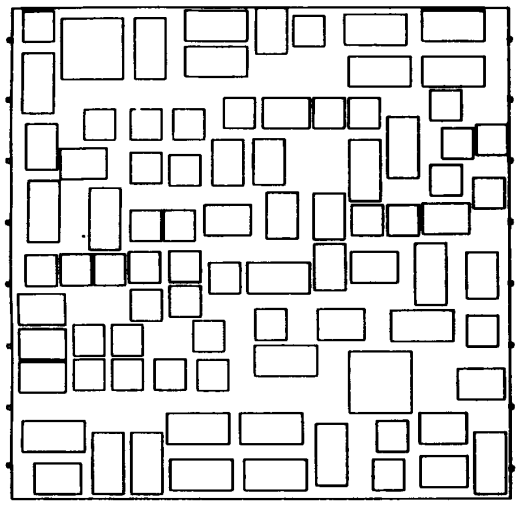

Figure 6. Placement of circuit 8.

8. Conclusion

In this paper, we presented a unique and effective approach to the performance driven macro–cell placement problem by translating both the timing and physical requirements into a set of upper– and lower–bounds on net wire lengths. We then presented a modified min–cut algorithm which tries to generate placements that satisfy as many of the upper– and lower–bounds as possible. We then discussed how the upper– and lower– bounds are to be modified to improve the timing performance of the placement. Experimental result are very encouraging.

References

[1] R. B. Hitchcock, G. L. Smith, and D. D. Cheng, "Timing Analysis of Computer Hardware," *IBM Journal of Research and Development*, 1982, vol. 26, no. 1, pp. 100–105.

[2] M. Burstein and M. N. Youssef, "Timing Influenced Layout Design," *Proc. 22th Design Automation Conference*, 1985, pp. 124–130.

[3] W. E. Donath et. al., "Timing Driven Placement Using Complete Path Delays," *Proc. 27th Design Automation Conference*, 1990, pp. 84–89.

[4] M. A. B. Jackson and E. S. Kuh, "Performance–Driven Placement of Cell Based IC's," *Proc. 26th Design Automation Conference*, 1989, pp. 370–375.

[5] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "Ritual: A Performance Driven Placement Algorithm for Small Cell ICs," *Proc. ICCAD*, 1991, pp. 48–51.

[6] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit Placement for Predictable Performance," *Proc. ICCAD*, 1987, pp. 88–91.

[7] A. E. Dunlop et. al., "Chip Layout Optimization Using Critical Path Weighting," *Proc. 21st Design Automation conference*, 1984, pp. 133–136.

[8] M. Marek–Sadowska and S. P. Lin, "Timing Driven Placement," *Proc. ICCAD*, 1989, pp. 94–97.

[9] Y. Ogawa et. al., "Efficient Placement Algorithms Optimizing Delay for High–Speed ECL Masterslice LSI's", *Proc 25rd Design Automation Conference*, 1986, pp. 404–410.

[10] W. M. Dai et. at., "BEAR: A New Building–Block Layout System," *Proc. ICCAD*, 1987, pp. 34–38.

[11] T. Gao, P. M. Vaidya, and C. L. Liu, "A New Performance Driven Placement Algorithm," *Proc. ICCAD*, 1991, pp. 44–47.

[12] S. Teig, R. L. Smith, and J. Seaton, "Timing Driven Layout of Cell–Based IC's," *VLSI System's Design*, pp 63–73, May 1986.

[13] M. Terai, K. Takahashi, and K. Sato, "A New Min–Cut Placement Algorithm for Timing Assurance Layout Design Meeting Net Length Constraint," *Proc. Design Automation Conference*, 1990, pp. 96–102.

[14] P. M. Vaidya, A New Algorithm for Minimizing Convex Functions Over Convex Sets," *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, 1989, pp. 332–337.

[15] C. M. Fiduccia and R. R. Mattheyses, "A Linear–Time Heuristic for Improving Network Partitions," *Proc. 19th Design Automation Conference*, 1982, pp. 175–181.