Quadratic Placement Using an Improved Timing Model

Bernd Obermeier
Institute of Electronic Design Automation
Technical University of Munich
80290 Munich
first_name.last_name@ei.tum.de

Frank M. Johannes
Institute of Electronic Design Automation
Technical University of Munich
80290 Munich
first_name.last_name@ei.tum.de

ABSTRACT

The performance of timing-driven placement methods depends strongly on the choice of the net model. In this paper a more precise net model is presented that does not increase numerical complexity. We introduce a method that replaces the clique model of a net by a tree model in the quadratic placement formulation. This improvement enables us to control the length of every tree segment separately. Furthermore, we present an analysis of the effects of every tree segment to the net delay. The result is in turn used to control the placement engine.
Our presented results are based on legal placements. They show significant improvements over state-of-the art methods.

Categories and Subject Descriptors

J.6 [**Computer-Aided Engineering**]: Computer-aided design (CAD)—*Physical Design, Layout Synthesis, Placement*

General Terms

Algorithms, Design

Keywords

timing driven placement, Quadratic placement, Steiner tree net model, sensitivity, optimization potential

1. INTRODUCTION

Advances of the production process of integrated circuits lead to shrinking feature sizes. This has various impacts on the physical behaviour of the logic devices and the interconnect. Gate delay decreases more rapidly than wire delay due to shrinking feature sizes [7]. Nets are responsible for an increasing percentage of total delay of modern designs. Together with the increasing importance of high performance designs, the ability to perform timing-driven placement has become a crucial task for today's EDA tools.
In general, approaches for the automated cell placement can be classified based on their fundamental algorithm. Recent publications regarding timing-driven placement include simulated annealing (SA) driven approaches, partitioning-based methods and adaptions of quadratic cost function algorithms.
An early SA-based approach was presented in [14], where the cost function was composed of half rectangle perimeter wirelength and a timing penalty. The netlist was clustered into two hierarchy levels and the placement process consisted of the interchange of clusters from the same hierarchy level, followed by swapping single cells. This approach has the ability to find the global optimum at the cost of very long run times. It has been used successfully for small designs. An advancement of this method is presented in [16]. The cluster stage has been replaced by a four-way hierarchical partitioning and the cost function is based on slack computation.
The authors of [12] propose a four-way partitioning approach. Timing-requirements are met by restricting the number of times a critical net is cut. Approach [10] consists of two stages: Netweights and a preliminary placement are computed based on slack minimization by linear programming in a first step. Afterwards, these results are used to guide a partitioning-driven placement engine.
Algorithms which use the squared lengths of all nets as cost function have experienced many variations lately. They differ in how they distribute the cells on the placement area and — in timing-driven placement context — in how they meet the timing constraints. A new approach for both meeting timing requirements and spreading the cells over the placement area was presented in [3]. Connecting the flip-flops of the start and end point of a critical path with pseudo nets is used to meet timing constraints. Cells are mapped to the rows by iteratively moving one cell at a time towards their final position. Other approaches like [13] and [5] used net weights to meet timing requirements. Quadratic placement tends to arrange cells connected with high weighted nets proximate to each other. One additional force working on each cell to distribute the cells evenly across the placement area was used in [5], where [13] iteratively introduced center of gravity constraints to spread the cells over the placement area. Center of gravity constraints to distribute the cells over the placement area were also used by [7], the limitation of the bounding box of critical nets by linear constraints ensures that timing specifications are met. This has been extended in [9] by adding some techniques to map the cells to their final position.

In this paper we propose a method which is based on the quadratic placement formulation. We use net weights to meet the timing constraints. Our technique can be integrated into center-of-gravity-based placement algorithms as well as into force-directed placement algorithms. It allows a more thorough analysis of the net delay and therefore a more precise control of the placement engine in order to meet timing requirements.

DAC 2004, June 7–11, 2004, San Diego, California, USA.

In detail, our approach implements the following enhancements to timing driven quadratic placement:

- We introduce a technique to incorporate a Steiner tree net model instead of the simple clique model into the quadratic cost function. The length of every tree segment can then be controlled individually.
- Based upon the Steiner tree net model we perform a *sensitivity analysis* of the net delay: For every Steiner tree segment, we compute the net delay derivative with respect to the segment length. This enables us to shorten preferably these segments of the Steiner tree which influence net delay most.
- The *optimization potential* of a Steiner tree segment gives us the net delay reduction when this segment is shortened to zero length. This enables us to predict how meaningful it is to further shorten a segment.

The rest of the paper is organized as follows: In section 2 we give a description of the timing- driven placement task. Section 3 reviews the timing-driven quadratic placement method and the Elmore delay computation used to adapt the net weights. This is followed by the description of our proposed method in section 4. Section 5 summarizes the results of sections 3 and 4 by presenting the improved timing-driven placement algorithm. The conducted experiments show excellent results, which are given in section 6. Section 7 concludes the paper.

2. PROBLEM FORMULATION

We describe the basis of the timing-driven placement problem in this section. First, we introduce the input data and give some definitions used in this paper later.

The input data consists of a directed graph $G(V, E)$ which represents the connectivity of the netlist. The node set V decomposes into two disjoint subsets V_c and V_p. V_c is the node set associated with combinational cells. Elements of V_p are associated with pads (primary inputs and primary outputs) and sequential cells (flip-flops). For simplicity we refer to every element of V as cell throughout this paper. For every combinational cell its propagation time (time that is needed for a value change at an input to be observable at its output) is known. A directed edge $e = (a, b) \in E$ from node $a \in V$ to node $b \in V$ exists in Graph G, if the output of node a drives an input of node b. Furthermore, additional informations are available like geometry data of all cells and the placement area which enable us to generate a legal placement.

A path P is a subgraph of G where the terminal node of an edge is identical with the first node of the next edge of P, the first and last nodes are from the set V_p and all other nodes are from the set V_c. The delay of a path is the time that is needed for a value change at the first node of the path to reach the last node. Common delay models compose the path delay of cell propagation delay and of the delay of the participating nets. During this work, we approximate the net delay with the Elmore delay [6], which depends on the placement of the cells and the net geometry. The complete path delay is therefore dependent on the placement of the cells, since net delay is responsible for a high percentage of the path delay. The longest path of a design is the path with the longest delay.

The task of timing-driven quadratic placement is to determine a global placement, where either the longest path is shorter than a certain threshold or the longest path has to be minimized in general. A global placement is characterized by an even distribution of all cells within the placement area, whereby the cells need not take their final positions, i.e. need not be mapped to rows. This will be accomplished by a final placer.

In this paper, we introduce significant technical progress to the timing driven quadratic placement approach, which will be reviewed in the next section.

3. TIMING-DRIVEN QUADRATIC PLACEMENT

This section addresses the conventional timing-driven quadratic placement method and pays special attention to the items subject to further development in the subsequent sections. The first part of this section deals with the plain quadratic placement method and the second part of this section gives details about the calculation of the net delay and its influence to the quadratic placement engine.

3.1 Quadratic placement revisited

Quadratic placement has its name from the structure of the cost function it seeks to minimize: The cost function is the weighted sum of all net lengths. The length of a two-pin net is defined as the squared Euclidean distance between these two pins. Multi-pin nets can be modeled as a clique. Therefore, a net with weight w connecting two pins i and j with coordinates (x_i, y_i) and (x_j, y_j) contributes to the cost function with $w \cdot ((x_j - x_i)^2 + (y_j - y_i)^2)$. The cost of a placement can be expressed as

$$K = \sum_{e=(i,j)\in E} w_e \cdot ((x_j - x_i)^2 + (y_j - y_i)^2),$$

where w_e is the weight of net e. If we collect the positions of all k movable cells of a design in a vector $\vec{p} = (x_1, x_2, \ldots, x_k, y_1, y_2, \ldots, y_k)$, the cost of a placement can be expressed as a quadratic form:

$$K = \frac{1}{2}\vec{p}^T \mathbf{C}\vec{p} + \vec{p}^T\vec{d} + const \qquad (1)$$

Matrix $\mathbf{C}$, vector $\vec{d}$ and the constant part represent the structure of graph G: a net connecting two movable cells contributes to Matrix $\mathbf{C}$, a net connecting a movable and a fixed cell contributes to both Matrix $\mathbf{C}$ and vector $\vec{d}$. Connections between two fixed cells add a constant value to the cost function.

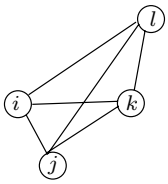
Due to the ordering of vector $\vec{p}$ matrix $\mathbf{C}$ decomposes into $\mathbf{C} = \begin{pmatrix} \mathbf{C_x} & 0 \\ 0 & \mathbf{C_y} \end{pmatrix}$, i.e. the x and y coordinates are independent. For the sake of simplicity we will treat only $\mathbf{C_x}$ and use $\mathbf{C}$ and $\mathbf{C_x}$ interchangeably for the rest of this section. $\mathbf{C_y}$ can be computed analogously. Every connection with weight w between two movable cells i and j (with positions (x_i, y_i) and (x_j, y_j), respectively) adds w to the matrix diagonal elements c_{ii} and c_{jj} and subtracts w from the off-diagonal elements c_{ij} and c_{ji}. A connection between a movable cell i and a fixed cell j contributes with w to matrix element c_{ii} and with $-w \cdot x_j$ to the ith element of vector $\vec{d}$.

Since every net connects only a small fraction of all pins, matrix $\mathbf{C}$ is sparse, but an n-pin net yields a dense $n \times n$ submatrix $\mathbf{C}'$ in $\mathbf{C}$. Fig. 1 shows a four pin net with weight w and the corresponding submatrix $\mathbf{C}'$.

The unique placement (positions of movable cells) with minimum cost is determined by the derivative of (1):

$$\mathbf{C}\vec{p} + \vec{d} = \vec{0} \qquad (2)$$

This placement is solely dependent on the positions of preplaced elements and the interconnect structure. Since a typical design has only few preplaced cells, the placement of



$$\mathbf{C}' = \begin{pmatrix} 3w & -w & -w & -w \\ -w & 3w & -w & -w \\ -w & -w & 3w & -w \\ -w & -w & -w & 3w \end{pmatrix}$$

Figure 1: Four pin clique model

minimum cost is characterized by strong cell overlaps and can serve only as a basis for further refinement steps.
Basically, two methods to distribute the cells evenly over the placement area have emerged in the past. The first technique partitions the set of cells and the placement area hierarchically. Every cell subset receives a center of gravity constraint corresponding to its part of the placement area. Prominent representatives of this technique are [15] and [11]. A more recent method to spread the cells within the placement area is the introduction of additional forces, first described in [5]. Every movable cell receives an additional force which guides the cells from high cell density areas to regions with low cell density. Regardless of the technique to spread the cells within the placement area, high-weighted nets tend to keep their connected cells closer together than low-weighted nets. Cell spreading takes place in an iterative fashion: Dependent on the actual placement, either a new partitioning level is constituted or the additional forces are updated in each iteration.
The net delay computation as described in the subsequent subsection serves as basis for the computation of the net weights. We will describe the complete conventional timing-driven quadratic placement method at the end of this section.

3.2 Net delay calculation revisited

In this subsection we describe the computation of the net delay, when the positions of the involved cells are given. We employ Elmore delay [6] to approximate the net delay, since it is easy to compute and is adequately correlated with the physical net delay.
For delay computation we need to take a closer look at the structure of a net: In general, an n-pin net consists of one net driver pin and $n-1$ load pins (the input pins of the driven cells). The computation of Elmore delay requires the knowledge of the internal resistance of the net driver pin and the input capacitances of the load pins. We further need the position of the pins, the resistance per unit length (denoted by r') and the capacitance per unit length (c') of the interconnect, and the interconnect geometry. Since delay is defined between two pins, the term "net delay" implicitly relates to the delay between the net driver pin and a specific target load pin. Figure 2 shows a Steiner tree as a model for the interconnect geometry. The dotted lines represent a grid which is spanned by the nodes corresponding to pins. When using Hanan's heuristic [8] to construct the Steiner tree, the tree segments run solely on this grid. We will use this figure as fundamental example for the rest of the paper. Node i corresponds to the net driver pin, pins j, k and l correspond to the load pins. The Steiner nodes are labeled s_1 and s_2. All tree nodes are collected in the set N. We explain the computation of the Elmore delay when pin k is the target load pin. Since Elmore delay requires a distinction between tree segments which are on the direct path from the net driver pin to the target load pin and tree segments which connect to the rest of the load pins, the former tree segments show up bold in figure 2 and the latter are drawn in gray. The lengths of the tree segments are given by l_{s_1} to l_k. We assign a direction to every tree segment which corresponds to the signal flow. Every tree segment is associated

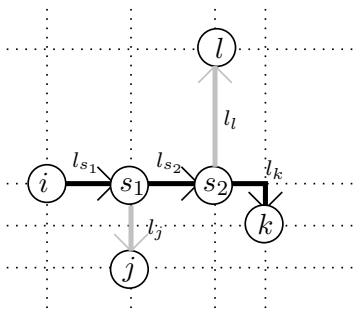


Figure 2: Steiner tree Example

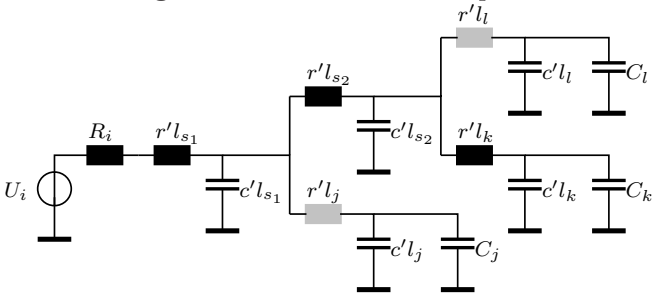


Figure 3: Elmore delay computation

with its end node, e.g. the segment of node k has length l_k. We denote the set of nodes on the path from a node p to r as $(p \dots r)$, whereas node p is not included in this set. Figure 3 shows the equivalent circuit diagram, where every tree segment is modeled as RC-element. Elmore delay neglects the impact of resistors which are not part of the path from the net driver pin to the target load pin. These resistors are also drawn in gray in figure 3. U_i is the voltage source, R_i is the internal resistance of pin i, and C_j, C_k and C_l are the input capacitances of the pins j, k and l. The capacitance of the driver pin is neglected. The capacitance assigned to a node is the length of its tree segment times c' plus the input capacitance of its corresponding pin. We assign a pin capacitance of 0 to all Steiner nodes, which makes the distinction between pin nodes and Steiner nodes obsolete. Every tree node contributes a portion to the Elmore delay from the net driver pin to the target load pin. The delay portion of an arbitrary node p is composed of the node capacitance times all resistances which are part of the intersection of the paths from the net driver node to both node p *and* the target node, i.e. we consider only the black-drawn resistors of figure 3. Let us consider the delay from node i to k as an example. The tree segments relevant for the delay portion of node l correspond to nodes $(i \dots l) \cap (i \dots k) = \{s_1, s_2\}$. The delay portion of node l is therefore $(r'(l_{s_1} + l_{s_2}) + R_i)(c'l_l + C_l)$. Please note, that $r'l_l$ does not show up, since it is not part of both paths from i to k and i to l. The Elmore delay D_k from the net driver node i to node k is the sum over the delay portions of all tree nodes as defined above:

$$D_k = \sum_{n \in N} (R_i + r' \sum_{\substack{m \in (i \dots n) \\ \cap (i \dots k)}} l_m) \cdot (c' l_n + C_n) \qquad (3)$$

This formula enables us to compute the signal delay from the net driver node to every load node when the net geometry is given. We use it to compute the delay of the longest path as introduced in section 2. An arrival time is assigned to the pins of every gate: If the pin is the output a gate, this is the maximum of the arrival times of the input pins plus the gate propagation time. The arrival time of an input pin is the net driver pin arrival time plus the net delay.

Initially, the arrival time of all elements of V_p is set to zero, the arrival times of the elements of V can then efficiently be computed by breadth-first searches starting from every element of V_p. The computation stops when reaching an element of V_p. The arrival times of these nodes directly give the path delays.
Usually, due to a desired clock frequency a required time for every path is specified by the designer. Similar to the arrival times, we can compute required times for all pins of the design: We initialize the required time of every node of V_p to its specified value (which is identical for all nodes of V_p in general). The required time of an input pin is the required time of the gate output pin minus the gate propagation time. The required time of a gate output pin is the minimum required time among all remaining pins of its net minus the corresponding net delay. The difference of the required time and the arrival time is called the slack of a pin. A negative slack of a output pin means, that its gate is on a path which violates the timing specifications. On the other hand, gates with positive slacks on its output pins are allowed to spread even further apart without a timing violation. Therefore the slack is used to control the net weights: The weight of a net connecting gates with negative slack will be increased in order to tighten the cells. Nets connecting cells with positive slacks will receive a decreased weight to enable its cells to spread further apart.
Algorithm 1 sketches the conventional timing-driven quadratic placement. The next section introduces our improvements

Algorithm 1 Timing-driven quadratic placement

```
Initialize C and d from netlist
Initialize all net weights with 1
repeat
  Compute updated placement (e.g. force-directed)
  Perform timing analysis:
    Compute arrival times, required times and slacks
  Adjust net weights based on slacks and set up C from
  scratch
until Convergence
```

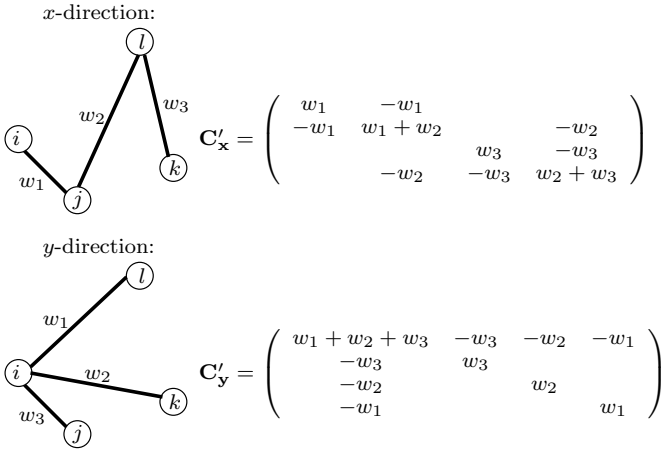
to the algorithm.

4. THE IMPROVED TIMING MODEL

This section gives a detailed description of our extensions to the conventional timing-driven quadratic placement flow. It is subdivided into three parts. The first part describes how to incorporate a Steiner tree net model into the cost function as a replacement for the clique model. The second and third part explain the sensitivity and optimization potential analysis as outlined in section 1.

4.1 Steiner tree net model

Choosing Hanan's heuristic [8] to construct the Steiner tree has several advantages: It is easy to compute and can potentially serve as global routing (the net model used for placement should naturally match the final routing as much as possible). Furthermore, Steiner nodes exist only on grid points spanned by the nodes corresponding to pins. This property preserves the dimension of matrix $\mathbf{C}$, as will be shown below.
When using a tree instead of a clique as net model, the cost function should be the weighted sum of the squared Euclidean lengths of all tree segments. Consider the segment between the nodes s_2 and k in figure 2: This edge adds one portion proportional to $(x_k - x_{s_2})^2$ and another portion proportional to $(y_k - y_{s_2})^2$ to the cost function. The coordinates of Steiner nodes in a tree set up with Hanan's heuristic can always be expressed by pin coordinates. Node s_2 in figure 2 has the coordinates $(x_{s_2}, y_{s_2}) = (x_l, y_i)$ since s_2 is directly below node l and on the same height as node i. Therefore we can construct two graphs to model the matrix entry scheme from one Steiner tree: One graph models the matrix entries for $\mathbf{C_x}$ and the other one the entries for $\mathbf{C_y}$. Referring to the example above, the segment between nodes s_2 and k in figure 2 results in one edge between nodes l and k in the graph corresponding to $\mathbf{C_x}$ and one edge between the nodes i and k in the graph corresponding to $\mathbf{C_y}$. The left half of figure 4 shows the graphs for the $\mathbf{C_x}$ matrix (top), resp. for the $\mathbf{C_y}$ matrix (bottom) resulting from the Steiner tree of figure 2. This approach enables us further to assign a distinct weight to every tree segment in both x- and y direction. For a better readability we gave the same indices to the edge weights in both graphs of figure 4, however the weights for $\mathbf{C_x}$ and $\mathbf{C_y}$ are certainly independent. The right part shows the matrices $\mathbf{C_x}$ and $\mathbf{C_y}$ resulting from the above graphs when each edge is treated as a two pin connection and entered as described in subsection 3.1. The row and column orders of the matrices are i, j, k, l. Please note, that the conventional timing driven quadratic placement algorithm also had to set up the matrix in every iteration, since the net weights change in every iteration. The additional cost of our appoach is solely caused by the construction of the Steiner trees. If the tree topology of one net changes at a certain iteration, the edge weights become invalid. The edge weights of the new tree model can be computed by the equivalence of the force working on each pin of the net exerted by the segment of the old tree and the new tree.
After we carried out how to incorporate a tree model [1] with separate weights for each segment into the cost function, we go on with the description of the optimization potential and sensitivity analysis in the next subsections. We will use these results in the following section to refine the net weighting procedure.



$$\mathbf{C'_x} = \begin{pmatrix} w_1 & -w_1 & & \\ -w_1 & w_1+w_2 & & -w_2 \\ & & w_3 & -w_3 \\ & -w_2 & -w_3 & w_2+w_3 \end{pmatrix}$$

$$\mathbf{C'_y} = \begin{pmatrix} w_1+w_2+w_3 & -w_3 & -w_2 & -w_1 \\ -w_3 & w_3 & & \\ -w_2 & & w_2 & \\ -w_1 & & & w_1 \end{pmatrix}$$

Figure 4: Graphs for matrix entries

4.2 Optimization potential analysis

As an introductory example, suppose the pins i and k of figure 2 have negative slacks, i.e. they violate the timing constraints, while pins l and j have positive slacks. Further suppose lengths l_{s_1}, l_{s_2} and l_k are much smaller (in the extreme case pins i and k abut) compared to lengths l_l and l_j. Although pins l and j do not violate the tim-

[1]This technique can also be useful in non timing-driven context: The bounding box of a net is determined by at most 4 pins. It is easy to identify the segments of the tree which influence the half rectangle perimeter of the net, whose weights are in turn increased.

ing constraints, shortening their segments is the only way to improve the slack values of pins i and k, since it will decrease the capacitance the net driver i has to charge and therefore decrease the net delay from pin i to pin k. This example demonstrates, that it is not sufficient to base the decision which net segment to shorten (and thus increasing its weight) solely on the slacks. We will give an expression for the portion of the net delay originating from a specific net segment m by extracting all terms which contain its segment length from formula 3. This expression gives directly the decrement of net delay when shortening this segment length to zero. We will denote this net delay decrement as optimization potential with respect to segment z. Formula 4 relates to the decrement of net delay from pin i to pin k (cf. figure 2) with respect to a segment z:

$$D_k - D_k|_{l_z=0} = c' l_z \cdot (R_i + r' \sum_{\substack{m \in (i \dots z) \\ \cap (i \dots k)}} l_m) + \begin{cases} r' l_z \sum_{m \in TS(z)} (c' l_m + C_m), \text{if } z \in (i \dots k) \\ 0, \text{ otherwise} \end{cases} \quad (4)$$

The set of transitive successors of node z is denoted by $TS(z)$ in formula 4. The first part of the formula refers to the delay caused by charging capacitor $c' l_z$, while the second part refers to the delays originating from charging downstream capacitors over resistor $r' l_z$. This part is only present, if the segment z is on the direct path from the net driver node to the target node, i.e. on the black drawn path of figure 2.

4.3 Sensitivity analysis

In this section we will derive a formula describing the variation of the net delay with respect to the length change of a tree segment z, i.e. $\frac{\partial D_k}{\partial l_z}$. Formula 4 directly gives the dependency of D_k from l_z. Therefore it holds true: $\frac{\partial D_k}{\partial l_z} = \frac{\partial (D_k - D_k|_{l_z=0})}{\partial l_z}$, which in turn yields:

$$\frac{\partial D_k}{\partial l_z} = c' \cdot (R_i + r' \sum_{\substack{m \in (i \dots z) \\ \cap (i \dots k)}} l_m) + \begin{cases} r' c' l_z + r' \sum_{m \in TS(z)} (c' l_m + C_m), \text{if } z \in (i \dots k) \\ 0, \text{ otherwise} \end{cases} \quad (5)$$

Please note, that a term $c' r' l_z^2$ shows up in the first part of formula 4 if segment z is on the direct path from the net driver node to the target node. This is considered by the first term of the second part of formula 5.

If we assume a Steiner tree composed of segments with roughly the same lengths we can interpret formulas 4 and 5. We then expect the longest segments on the path from the net driver pin to the target load pin to have both the greatest optimization potential and the greatest relative influence to the net delay. Hence we can give the outline of our net weighting strategy for nets violating the timing constraints (i.e. nets which contain pins with negative slack). If there are long Steiner tree segments on the path from the net driver node to the node with the smallest slack, we will increase their weights. In this context, a long Steiner tree segment relates to a segment with a great optimization potential. Since these segments strongly influence the net delay, a moderate weight increment is sufficient to meet timing constraints and preserves the maximum degree of freedom for the placement engine. We have to increase the net weights of the remaining segments more strongly. The details about the interaction between slack, optimization potential and sensitivity are given in the following section.

5. IMPLEMENTATION

The first part of this section gives a descriptive explanation of the original slack-based net weighting procedure, while the second part shows the strategy we chose to incorporate the optimization potential and sensitivity information into the net weighting procedure.

During every iteration of the placement algorithm the net weights are recomputed according to the actual placement. We use the placement information to compute the arrival times, required times and slacks for every pin, as outlined in section 3.2. We define the net slack to be the minimum slack of its pins, since contracting any part of the net will improve the minimum slack among all pins. During every iteration of the algorithm we determine the net weight increment as the product of the old net weight and a slack-based weight factor. This factor is 0 if the slacks of a net were 0 during the past iterations, since this net precisely meets the timing constraints while its present weight offers the greatest degree of freedom for the placement algorithm. The weight factor increases with the amount of timing violations and decreases if the net slack is positive.

To make use of the improved net model introduced in the previous sections, we have to compute a weight factor for each segment of the Steiner tree separately. The weight factors are based on the computation described above and altered to account for the optimization potential and the sensitivity of each segment. The variable g_o denotes the optimization potential (formula 4) and g_s denotes the net delay sensitivity (formula 5) with respect to one segment. Both variables are scaled to the maximum optimization potential (resp. sensitivity) of the net. The weight factor for each segment is multiplied with $\alpha g_o \cdot g_s e^{-\beta g_s}$. The constants α and β balance the influence of the optimization potential and sensitivity versus the slack and the force field. They are determined by experiment. This strategy features the following properties:

- No weight change takes place, if the segment has no optimization potential.
- The larger the optimization potential of one segment is, the more it is used to meet the timing constraints.
- If the sensitivity of the net delay with respect to one segment length is zero, this segment keeps its weight.
- With increasing sensitivity the weight factor increases until it reaches a maximum. This accounts for the lower sensitivity of these segments which are not on the direct path from the net driver pin to the target load pin.
- Segments on the direct path from the net driver pin to the target load pin receive a lower weight increment, due to their higher sensitivity.

The absolute minimum and maximum edge weights are bounded as well as their ratio. The bounds have been determined experimentally. We integrated our novel net model and net weighting approach into the force- directed quadratic placement engine described in [5]. Algorithm 2 summarizes our procedure. The enhancements to the net weight computation resulted in significant improvement of the longest path delay as will be shown in the next section.

6. RESULTS

We used the subset of the MCNC benchmarks [1] which contains timing information to compare our approach with previous works. In order to enable a fair comparison with the original approach, the conducted experiments consist of

Algorithm 2 Improved timing-driven quadratic placement

Initialize $\mathbf{C}$ and $\vec{d}$ from netlist
Initialize all net weights with 1, force vector $\vec{e} = \vec{0}$.
repeat
  Compute updated placement $\mathbf{C}p + \vec{d} + \vec{e} = \vec{0}$.
  Set up Steiner tree.
  Perform timing analysis:
    Compute arrival times, required times and slacks.
  Compute weight factor based on net slack.
  Compute optimization potential and sensitivity for every net segment.
  Adjust segment weights based on slacks, optimization potential and sensitivity and set up $\mathbf{C}$.
  Compute force field and update $\vec{e}$.
until Convergence

circuit	lower bound [ns]	Eisenmann [5] [ns]	this work [ns]	rel. improvement
fract	18.5	19.8	19.7	8 %
struct	84.0	90.1	89.1	16 %
biomed	27.0	35.7	33.2	29 %
avq.small	69.9	80	76.9	31 %
avq.large	79.9	94	86.9	50 %
average				27 %

Table 1: Longest Path

global placement runs as described in the previous sections followed by final placement steps. The final placement steps align the cells in rows and provide a legal placement. We used Domino [4] as final placer. We compare our approach with [5] which is the latest work that presented legal placement results for publicly available benchmarks. More recent publications (like [7] and [12]) using publicly available benchmarks only presented global placement results without any information about the amount of remaining overlap. Other works like [16], [10], and [9] used proprietary benchmarks to present results. Table 1 shows the comparison of [5] with our approach. The first column gives the delay of the longest path when disregarding the net delay, i.e. the sum of the cell delays on the longest path. This number marks the lower bound for timing optimization. The second and third column show the delay of the longest paths after final placement. To compute these numbers, we assumed a capacitance per length of 242 pF/m and resistance per length of 25.5 kΩ/m. The last column shows the delay improvement relative to the maximal possible improvement. The delay improvement is the difference of Eisenmann's approach and this work. The maximal possible improvement is the difference of Eisenmann's approach and the lower bound. Thus, the last column gives the ratio of the delay improvement and the maximal possible improvement. We were able to achieve an average relative improvement of 27 percent over [5]. This is the first time that results are published which show improvements over Eisenmann's placement algorithm when comparing legal placements.

The numerical complexity of our approach dependent on the number of placeable cells is comparable to [5], since the proposed improvements affect every net once per iteration and the average number of pins per net does not depend on the number of placeable instances. With the cpu-time of [5] scaled according to [2], our approach is roughly 25 % slower than [5] over all benchmarks.

7. CONCLUSION

Decreasing feature sizes of the production process of integrated circuits increase the net delay portion of the total delay. The rising importance of high performance designs demands the development of more powerful placement techniques to minimize the delay of the longest path.

In this work, we introduced significant improvements to the timing-driven quadratic placement methodology. We described the incorporation of a Steiner tree model into the quadratic cost function and derived expressions for the optimization potential and the sensitivity of the net delay with respect to a tree segment. These informations have been used to control the placement engine which in turn led to significant improvements over the state-of-the art methods.

8. REFERENCES

[1] Benchmarks. http://www.cbl.ncsu.edu/.

[2] The performance database server. "performance.netlib.org/performance/html/PDStop.html".

[3] Y.-C. Chou and Y.-L. Lin. A performance-driven standard-cell placer based on a modified force-directed algorithm. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 24–29, 2001.

[4] K. Doll, F. M. Johannes, and K. J. Antreich. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 13(10):1189–1200, Oct. 1994.

[5] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *ACM/IEEE Design Automation Conference (DAC)*, pages 269–274, June 1998.

[6] W. C. Elmore. The transient response of damped linear networks with particular regard to wide-band amplifiers. *J. Appl. Phys. 19*, pages 55–63, 1948.

[7] B. Halpin, C. R. Chen, and N. Sehgal. Timing driven placement using physical net constraints. In *ACM/IEEE Design Automation Conference (DAC)*, 2001.

[8] M. Hanan. On steiner's problem with rectiliner distance. *SIAM Journal of Applied Mathemetics*, 14(2):255–265, 1966.

[9] S.-W. Hur, T. Cao, K. Rajagopal, Y. Parasuram, and B. Halpin. Force directed mongrel with physical net constraints. In *ACM/IEEE Design Automation Conference (DAC)*, 2003.

[10] A. B. Kahng, S. Mantik, and I. L. Markov. Min-max placement for large-scale timing optimization. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, 2002.

[11] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, CAD-10(3):356–365, Mar. 1991.

[12] S.-L. Ou and M. Pedram. Timing-driven placement based on partitioning with dynamic cut-net control. In *ACM/IEEE Design Automation Conference (DAC)*, 2000.

[13] B. M. Riess and G. Ettelt. SPEED: Fast and efficient timing driven placement. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 377–380, Seattle, Washington, Apr. 1995.

[14] W. Swartz and C. Sechen. Timing driven placement for large standard cell circuits. In *ACM/IEEE Design Automation Conference (DAC)*, pages 211–215, 1995.

[15] J. Vygen. Algorithms for large-scale flat placement. In *ACM/IEEE Design Automation Conference (DAC)*, pages 746–751, 1997.

[16] X. Yang, B.-K. Choi, and M. Sarrafzadeh. Timing-driven placement using design hierarchy guided constraint generation. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2002.