# Fast and Robust Quadratic Placement Combined with an Exact Linear Net Model

Peter Spindler and Frank M. Johannes
Institute of Electronic Design Automation, Technische Universitaet Muenchen, Munich, Germany

*Abstract*— **This paper presents a robust quadratic placement approach, which offers both high-quality placements and excellent computational efficiency. The additional force which distributes the modules on the chip in force-directed quadratic placement is separated into two forces: hold force and move force. Both of these forces are determined without any heuristics. Based on this novel systematic force implementation, we show that our iterative placement algorithm converges to an overlap-free placement. In addition, engineering change order (ECO) is efficiently supported by our placer. To handle the important trade-off between CPU time and placement quality, a deterministic quality control is presented.**

**In addition, a new linear net model is proposed, which accurately models the half-perimeter wirelength (HPWL) in the quadratic cost function of quadratic placement. HPWL in general is a linear metric for netlength and represents an efficient and common estimation for routed wirelength. Compared with the classical clique net model, our linear net model reduces memory usage by $75\%$, CPU time by $23\%$ and netlength by $8\%$, which is measured by the HPWL of all nets.**

**Using the ISPD-2005 benchmark suite for comparison, our placer combined with the new linear net model has just $5.9\%$ higher netlength but is $16\times$ faster than APlace, which offers the best netlength in this benchmark. Compared to Capo, our placer has $9.2\%$ lower netlength and is $5.4\times$ faster.**

**In the recent ISPD-2006 placement contest, in which quality is mainly determined by netlength and CPU time, our placer together with the new net model produced excellent results.**

## 1. Introduction

As Moore's law is still valid [2], i.e. circuit sizes are doubled every eighteen months, new fast and efficient placement algorithms with accurate new models will be needed for the layout synthesis of next-generation VLSI circuits with tens of millions of standard cells.

State-of-the-art placers can be classified in three categories:
*(1) Simulated-Annealing Approaches*
Simulated-annealing provably find the global optimum but placers based on this optimization approach usually suffer from long run times. The best known representative of this placer category is Timberwolf [30].


ICCAD'06, November 5-9, 2006, San Jose, CA


*(2) Partitioning Approaches*
Another placement approach is to recursively partition the circuit and the placement area based on a certain cost function, e.g. number of wires crossing a boundary of adjacent partitions. Recent placers using this technique are Capo [28], Dragon [31], FengShui [5], and NTUPlace [9].
*(3) Analytical Approaches*
The core of all analytical placers is an objective function which is minimized by methods of mathematical analysis. Depending on the kind of objective function, analytical placers can be subdivided into two categories:
*(i) Nonlinear-Optimization-Based Placers:* The objective function is nonlinear, e.g. a log-sum-exponential function [25], which is minimized by nonlinear optimization techniques like conjugate-gradient optimization [22]. Examples of nonlinear-optimization-based placers are APlace [18] and mPL [7].
*(ii) Quadratic Placers:*The objective function is quadratic and can therefore be minimized efficiently by solving a system of linear equations. Quadratic placers are for instance Gordian [19], Kraftwerk [12], FAR [15], FastPlace [32], mFAR [16], BonnPlace [6], and hATP [24].

To cope with modern circuits having millions of modules, many placers combine different optimization techniques with a hierarchical approach, e.g. mFAR [16], Dragon [31], APlace [18], mPL [7], and hATP [24].

Quadratic placers are popular, because they allow good quality results at low CPU times. But they face two problems: First, a technique is needed to reduce the module overlap, which usually exists in quadratic placement. Depending on this technique, quadratic placers can be divided into two categories: (1) Constraint-based quadratic placers like [19],[6],[24], which achieve an overlap-free placement by center of mass constraints. (2) Force-directed quadratic placers like [12],[15],[32],[16], which distribute the modules on the chip by an additional force. The second problem of quadratic placers is that the quadratic objective function is just an approximation for the real objective, e.g. routed wirelength. Both problems are addressed in this paper: a systematic non-heuristic formulation of the additional force is presented and an exact method to express routed wirelength in the quadratic objective function is shown.

Different approaches appeared to implement the additional force needed in force-directed quadratic placement. Kraftwerk [12] utilizes module density to determine a constant additional force which drives the modules from high to low density regions. FAR [15] calculates the additional force like Kraftwerk but models it by fixed points. mFAR [16] uses two different fixed points to express the additional force. The

first "perturbing fixed points" reduce module overlap and are calculated heuristically by local bin utilization. The second "controlling fixed points" achieve force equilibrium and are determined also by heuristics. FastPlace [32] uses a similar technique for the additional force as mFAR [16].

In general, all quadratic placement approaches formulate nets by a clique net model or the equivalent star net model and use net weights for adaptation of the quadratic cost function to a realistic objective, e.g. linear netlength to express routed wirelength. With the number of pins denoted by $P$, a common weight for the clique net model is $1/P$ in order to adapt its cost function to the star net model [21], [32]. For additional linearization Vygen et al. [33] use a net weight of $1/(P-1)$ and Kleinhans et al. [19] set the net weight to $2/P$. These approximation techniques express linear netlength in a quadratic cost function in a heuristical manner [29].

In this paper we describe a fast, robust, flat, iterative force-directed quadratic placer together with a new linear net model. In detail our quadratic placer is characterized by the following enhancements to other quadratic placement approaches:

- We separate the additional force of force-directed quadratic placement into two fundamental components: move force and hold force:
  - We use the module density and the potential formulation of [12] to calculate a non-heuristic move force which is implemented by target points.
  - To improve the convergence of the iterative process of quadratic placement, we use a constant and non-heuristic hold force.
- Based on this new systematical force modeling, module positions can be computed efficiently. Moreover we prove that our quadratic placer converges to an overlap-free placement.
- As a result of the force separation, our placement algorithm can be easily restarted at any iteration without any initialization. This efficiently supports the engineering change order (ECO), where the circuit is slightly modified and is needed to be placed again.
- To control the important trade-off between CPU time and quality of placement, we implement a deterministic quality control with just one single parameter.
- In order not to narrow the design space, we do not utilize a hierarchical approach, but place all modules simultaneously in each iteration, i.e. our placer is flat.

The following properties distinguish our new linear net model from previous net models. Please note that our net model can be universally utilized by any quadratic placer.

- Exact and deterministic representation of the half-perimeter wirelength (HPWL) in a quadratic objective function. The HPWL is defined per net by the half-perimeter of the bounding box enclosing its pins. The HPWL represents a linear metric for netlength and moreover a common and efficient estimation for routed wirelength.
- Efficient removal of module overlap.
- Lower memory usage and runtime.

The rest of the paper is organized as follows: In section 2 we give a brief introduction to quadratic placement and we describe our placement approach in detail. After a short description of the traditional clique net model, our new linear net model is presented in section 3. Experimental results are provided in section 4, followed by the conclusion in section 5.

## 2. Quadratic Placement Methodology

A placement process in general aims at optimal places for modules with the objective of minimizing netlength under the constraint of no overlap between modules. To find minimal netlength, the connectivity between modules, as described in the netlist, is formulated in a netgraph with modules as vertices and nets as hyperedges connecting subsets of modules.

In quadratic placement only two-point connections can be used to express netlength, so a net model is necessary to transform the hyperedges into two-points connections. The transformation of the hyperedges into two-point connections results in a binary graph $(M, E)$ with a set of edges $E$ connecting pairs of modules in set $M$. In quadratic placement the weighted squared Euclidean length $\Gamma_e$ between two vertexes $i$ and $j$, respectively two modules $i$ and $j$ is assigned to each edge $e = (i, j) \in E$:

$$\Gamma_e = \frac{w_{e,x}}{2}(x_i - x_j)^2 + \frac{w_{e,y}}{2}(y_i - y_j)^2 = \Gamma_{e,x} + \Gamma_{e,y} \quad (1)$$

Each edge length $\Gamma_e$ adds to the cost function $\Gamma$ of quadratic placement:

$$\Gamma = \sum_{e \in E} \Gamma_e = \Gamma_x + \Gamma_y \quad (2)$$

Since $\Gamma_e$ can be separated in x- and y-dimension, the cost function $\Gamma$ can be separated in the same way. The following description for the x-dimension applies likewise for the y-dimension.

Splitting all modules in $N$ movable and $M$ fixed modules, the x-positions of movable modules can be collected in vector $\mathbf{x} = (x_1, x_2, x_3, ..., x_N)^T$ and the quadratic cost function $\Gamma_x$ can be written in matrix-vector notation [13]:

$$\Gamma_x = \frac{1}{2}\mathbf{x^T C_x x} + \mathbf{x^T d_x} + const \quad (3)$$

Here matrix $\mathbf{C_x}$ is of dimension $N \times N$ and has matrix entry $c_{ij}$ in row $i$ and column $j$. Vector $\mathbf{d_x}$ is of dimension $N$ and has entry $d_{x,i}$ in row $i$. To express the length $\Gamma_{e,x} = 0.5\, w_{e,x}(x_i - x_j)^2$ between two movable modules $i$ and $j$ in matrix-vector notation (3), the matrix entries $c_{ii}$ and $c_{jj}$ are increased by $w_{e,x}$ and the diagonal matrix entries $c_{ij}$ and $c_{ji}$ are decreased by $w_{e,x}$. If one module — let's say $j$ — is fixed then matrix entry $c_{ii}$ is increased by $w_{e,x}$ and vector entry $d_{x,i}$ is decreased by $w_{e,x} \cdot x_j$. The length between two fixed modules just contributes to the constant part of (3).

As the total netlength is expressed by the cost function $\Gamma$, the module positions for minimal netlength in quadratic placement are found by minimizing $\Gamma$. For x-dimension this is done by differentiating $\Gamma$ with respect to x and solving the resulting system of linear equations with respect to $\mathbf{x}$. Denoting the vector differential operator $\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_N}\right)^T$ by the nabla operator $\nabla_x$, the minimization of netlength requires the solution of:

$$\nabla_x \Gamma = \nabla_x \Gamma_x = \mathbf{C_x x} + \mathbf{d_x} = \mathbf{0} \quad (4)$$

Solving this system of linear equations for $\mathbf{x}$ can be done efficiently, e.g. by applying the conjugate-gradient technique, since matrix $\mathbf{C_x}$ is highly sparse.

Quadratic placement as formulated above can be compared with an elastic spring system: length $\Gamma_e$ of edge $e$ is equal to the spring energy $E = 0.5\,w\left(\Delta x^2 + \Delta y^2\right)$ with spring constant $w$ and squared Euclidean spring elongation $\Delta x^2 + \Delta y^2$. As derivative of energy is force and $\Gamma$ expresses the total netlength and therefore the total spring energy, the derivative of $\Gamma$ with respect to x is the total net force in x-direction:

$$\nabla_x \Gamma = \mathbf{F_x^{net}} = \mathbf{C_x x} + \mathbf{d_x} = \mathbf{0} \tag{5}$$

This net force is set to zero to find the minimum energy of the elastic spring system, which is equal to the minimum netlength.

With (4) and (5) the force created by all springs, respectively all connections, adjacent to module $i$ is expressed in $F_{x,i}^{net}$ being the entry in row $i$ of force vector $\mathbf{F_x^{net}}$.

### 2.1. Additional Forces

With just net forces acting on the modules, the modules attract each other resulting in a lot of module overlap. To reduce this overlap, additional forces are needed. Traditionally, the complete process of reducing module overlap is formulated as an iterative process where in each iteration the overlap is reduced.

In this paper, we represent the module positions from last iteration in vector $\mathbf{x}'$, the module positions calculated in the current iteration in vector $\mathbf{x}$, and the change in module position between two iteration in vector $\mathbf{\Delta x}$:

$$\mathbf{\Delta x} = \mathbf{x} - \mathbf{x}' \tag{6}$$

Furthermore we separate the additional force needed in quadratic placement into two fundamental components. First, a hold force which holds the modules in the current iteration and thus decouples the current iteration from the previous one. Second, a move force which moves the modules in the current iteration to reduce the module overlap.

#### 2.1.1. Move Force

To calculate the move force we employ the non-heuristic approach of [12] by formulating the placement problem as a global electrostatic problem: modules with positive charge and chip area with negative charge form a charge density $D$. This charge density creates a potential $\Phi$, which can be solved by the Poisson equation:

$$\triangle \Phi = -D \tag{7}$$

The Poisson equation can be solved efficiently by a geometric multigrid solver [27], [20].

If chip's charge density $D_{Chip}$ is constant then module density $d$ is exactly charge density $D$ plus $D_{Chip}$ [12]. Therefore, no difference between module density $d$ and charge density $D$ is made further on.

In the electrostatic formulation (7), the potential $\Phi$ is high in regions with high module density and low in regions with low module density. Since the negative gradient $\left(-\frac{\partial \Phi}{\partial x}, -\frac{\partial \Phi}{\partial y}\right)^T$ of potential $\Phi$ points in the direction of highest density decrease, this gradient can be used to move the modules from high density to low density regions in order to equalize the module density and therefore reduce the overlap between modules.

Hence each module $i$ gets a target point $\mathring{x}_i$ and is connected to its target point with a spring with the spring constant $\mathring{w}_i$. The target point is calculated by:

$$\mathring{x}_i = x_i' - \frac{\partial}{\partial x}\Phi\Big|_{(x_i', y_i')} \tag{8}$$

Each spring connection to a target point creates a move force $F_{x,i}^{move} = \mathring{w}_i(x_i - \mathring{x}_i)$ which is collected in the move force vector $\mathbf{F_x^{move}}$:

$$\mathbf{F_x^{move}} = \mathring{\mathbf{C}}_\mathbf{x}\left(\mathbf{x} - \mathring{\mathbf{x}}\right) \tag{9}$$

The matrix $\mathring{\mathbf{C}}_\mathbf{x}$ is a diagonal matrix with the spring constants as entries: $\mathring{\mathbf{C}}_\mathbf{x} = diag(\mathring{w}_i)$. Collecting the gradients of $\Phi$ in x-dimension for all modules in vector $\mathbf{\Phi_x}$

$$\mathbf{\Phi_x} = \left(\frac{\partial}{\partial x}\Phi\Big|_{(x_1', y_1')}, \frac{\partial}{\partial x}\Phi\Big|_{(x_2', y_2')}, ..., \frac{\partial}{\partial x}\Phi\Big|_{(x_N', y_N')}\right)^T \tag{10}$$

the target point vector $\mathring{\mathbf{x}}$ can be calculated by $\mathring{\mathbf{x}} = \mathbf{x}' - \mathbf{\Phi_x}$.

#### 2.1.2. Hold Force

If the hold force $\mathbf{F_x^{hold}}$ is defined by the negative net force

$$\mathbf{F_x^{hold}} = -\left(\mathbf{C_x x}' + \mathbf{d_x}\right) \tag{11}$$

and the sum of hold force and net force is set to zero, then the modules are held on their current positions, i.e. $\mathbf{x} = \mathbf{x}'$.
Proof:

$$\mathbf{F_x^{net}} + \mathbf{F_x^{hold}} = \mathbf{C_x x} + \mathbf{d_x} - \mathbf{C_x x}' - \mathbf{d_x} = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{x}' \tag{12}$$

■

Since all three components $\mathbf{C_x}$, $\mathbf{x}'$ and $\mathbf{d_x}$ of the hold force do not depend on $\mathbf{x}$, the hold force itself is constant.

#### 2.1.3. Total Force

The net force, move force and hold force add up to the total force $\mathbf{F_x}$. The total force is then set to zero to get a placement with minimal netlength and some overlap reduction:

$$\mathbf{F_x} = \mathbf{F_x^{net}} + \mathbf{F_x^{move}} + \mathbf{F_x^{hold}} = \mathbf{0} \tag{13}$$

Altogether our systematic non-heuristic force-directed quadratic placement approach differs significantly from other force-directed quadratic placement approaches [12], [15], [32], and [16], all of which either do not separate the additional force or use various heuristics to obtain the additional force. Contrary to that,

- we separate the additional force in hold force and move force,
- we calculate the move force (9) non-heuristically by an electrostatic potential (7) and model it by target points (8),
- we represent the hold force by a non-heuristic constant force (11).

This results in the simple formulation of the total force (13) of our placer by:

$$\mathbf{F_x} = \left(\mathbf{C_x} + \mathring{\mathbf{C}}_\mathbf{x}\right)\mathbf{\Delta x} + \mathring{\mathbf{C}}_\mathbf{x}\mathbf{\Phi_x} = \mathbf{0} \tag{14}$$

The new module positions $\mathbf{x} = \mathbf{x}' + \mathbf{\Delta x}$ in x-dimension are efficiently computed by solving (14) for $\mathbf{\Delta x}$. To obtain the module positions in y-dimension, all described steps must be executed for y-direction.

### 2.2. Proof of Convergence

Please note that in general, the placement problem is NP-hard and all placement approaches model the problem by algorithms which can be executed with polynomial time complexity [11]. In our placement approach we apply Poisson's equation (7) and the consequential force formulation (4), (9), (11), and (14). Our placer is unique because it does not resort to heuristics. Therefore it is robust and we can prove that our iterative global placement approach converges to an overlap-free placement.

Sketch of proof:

1) With no move force, the modules are held with hold force at their current position, which is proven in section 2.1.2.
2) The potential $\Phi$ represents the module density $D$ (7) and the negative gradient of the potential $\Phi$ is used to calculate the target points (8). So the move force, which is determined by the target points, moves the modules away from high density regions to low density regions (9).
3) Therefore the maximum module density is decreased and the minimum module density is increased in each iteration. This can be proven by using (7), (8), (9), (11), (14) and some theorems of electrodynamic theory. Due to the page limit, the complete description is omitted here and will be published in a future paper. Lowering the maximum module density and rising the minimum module density in each iteration equalizes the module density $D$ more, which means that module overlap is reduced.
4) If the density $D$ is totally equalized, which means that all overlap between modules is removed, then the modules do not move anymore (i.e. $\mathbf{\Delta x} = \mathbf{0}$) and therefore the global placement algorithm has converged to an overlap-free placement. This is true because for an equalized density $D$ the potential $\Phi$ is constant. Thus the gradient of $\Phi$ is zero and hence $\mathbf{\Phi_x}$ is zero. So (14) is transformed to $\left(\mathbf{C_x} + \mathring{\mathbf{C}}_\mathbf{x}\right)\mathbf{\Delta x} = \mathbf{0}$ and accordingly $\mathbf{\Delta x} = \mathbf{0}$.

■

Two remarks must be added to the proof of convergence:

- An assumption has to be made: two modules $i$ and $j$ may not have the same position: $(x_i, y_i) \neq (x_j, y_j)$. If they have exactly the same position then they will get the same move force and will probably be moved to the same position in the next iteration and hence the overlap between these two modules will not be removed. Practically this assumption has no impact on convergence since the module positions are calculated numerically and therefore will not be exactly the same.

  Even if two modules $i$ and $j$ have exactly the same position, the modules connected to these two modules $i$ and $j$ will probably move them to different positions in the next iteration.

**Initial Placement:**
Place all modules in the center of the chip
**for** $i < I_{init}$ **do**
For x-direction: (similarly for y-direction)
Create $\mathbf{C_x}$, $\mathbf{d_x}$
Solve (4) for $\mathbf{x}$
$i = i + 1$

**Global Placement:**
**repeat**
Calculate potential $\Phi$ by (7)
For x-direction: (similarly for y-direction)
Create $\mathbf{C_x}$, $\mathring{\mathbf{C}}_\mathbf{x} = diag(\mathring{w}_i)$, $\mathbf{\Phi_x}$
Solve (14) for $\mathbf{\Delta x}$
Update module position $\mathbf{x}$ by $\mathbf{\Delta x}$
Quality control
**until** Module overlap $\leq 20\%$

**Final Placement:**
FindNextBestPlace

Fig. 1. Complete placement algorithm

- No theoretical statement to the iteration count of global placement can be made as this highly depends on the kind of circuit to be placed. But experiments showed that a practical bound of iteration count is around 5 if a high spring constant (e.g. 1000) of the target points is chosen.

### 2.3. Implementation Details

Figure 1 shows the complete algorithm of our placer. During initial placement, a start solution is computed by minimizing cost function $\Gamma$ over a few iterations: $I_{init} \approx 5$. At this stage module overlap is not taken into account.

After that the module overlap is reduced iteratively in global placement. Although global placement converges to an overlap-free placement as proven above, it is stopped at a certain stopping criterion, e.g. module overlap $\leq 20\%$.

Global placement is terminated in standard-cell placement because it can not arrange the modules on the chip rows which is needed to obtain a legal placement. This task and removing the remaining module overlap is done during final placement by "FindNextBestPlace": the next best place is sought for each module according to a certain cost function. This search takes around $10\%$ of CPU time of the global placement and therefore is fast. Using netlength in HPWL metric as cost function increases total HPWL by around $2\%$ compared to the last iteration of global placement.

### 2.4. Engineering Change Order

The separation of the additional force in hold force and move force results in decoupling one iteration from the previous one. Therefore our global placement algorithm can be easily restarted at any iteration without special initializations. Thus the engineering change order (ECO) is efficiently supported. This means that after a small change in a circuit, e.g. after gate sizing, the circuit can be placed again without running the whole placement process from scratch, but starting immediately the placement algorithm from the last iteration.

### 2.5. Quality Control

In order to control the important trade-off between the quality of placement and the CPU time, a quality control procedure is called at the end of each iteration in global placement (see figure 1).

This trade-off presents a challenge in everyday placement usage. On the one hand the deadline for placement can be near and therefore the chip has to be placed in short CPU time. On the other hand the quality of placement can be very important with no limit of CPU time.

As described in section 1, every connection has a weight $w_e$. These connection weights are changed in each iteration of global placement in order to model a realistic objective, e.g. timing requirements [26], or routed wirelength, in the quadratic cost function. Hence the more iterations are spent in the global placement, the better is the modeling of the real objective, i.e. the higher is the quality. On the other hand, the more iterations the global placement needs, the higher is the CPU time, since every single iteration takes a fix CPU time.

If the average module movement $\mu$ is controlled to be $\mu_T$ in every iteration, then the iteration count $I_{global}$ of global placement is indirectly proportional to $\mu_T$: each module has to move a certain length $L$ in global placement in order to get an overlap-free placement and thus following holds true:

$$L = \mu_T \cdot I_{global} \Leftrightarrow I_{global} \propto \frac{1}{\mu_T} \tag{15}$$

To control the module movement $\mu$ to be $\mu_T$, the target points' spring constants $\mathring{w}_i$ are used, since a target point with a small spring constant attracts its module less than with a high spring constant.

So altogether, a certain target movement $\mu_T$ is set for quality control and the average module movement $\mu$ is compared to $\mu_T$ in each iteration: if $\mu < \mu_T$ then every $w_{T,i}$ is reduced and if $\mu > \mu_T$ then every $w_{T,i}$ is increased.

Please note that the introduced quality control does not affect the proof of convergence in section 2.2 since there is no statement about the target points' spring constant.

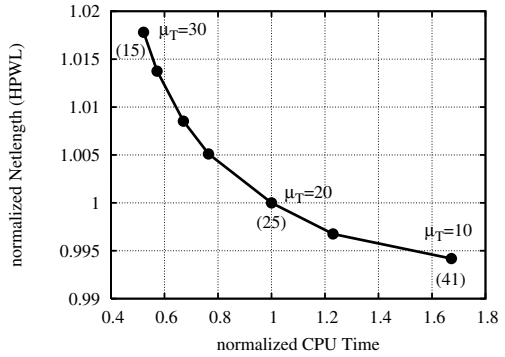



Fig. 2. Trade-off between quality of placement, measured in HPWL netlength, and CPU time with quality control's parameter $\mu_T$. The results are based on six circuits of the ISPD-2005 benchmark. The values in the brackets express the average of iteration count $I_{global}$.

Figure 2 shows that the presented quality control can efficiently govern the important trade-off between quality of placement and CPU time by using its only parameter $\mu_T$. Here the quality of placement is measured in the HPWL of all nets, where the HPWL expresses a linear netlength and moreover an efficient estimation for routed wirelength. Compared to $\mu_T = 20$, the CPU time can be decreased to $50\%$ at $\mu_T = 30$. At this point, the quality of placement is less than $2\%$ worse than at the starting point. On the other side at $\mu_T = 10$, the quality can be improved by around $0.5\%$ at a CPU time increase of $70\%$. With a quality range of less than $2\%$ and a CPU time range more than $100\%$, figure 2 also demonstrates that our placement algorithm is very robust in quality of placement but flexible in CPU time.

## 3. Linear Net Model

Section 2 describes that the netlist is modeled as a graph with modules as vertices and nets as hyperedges. Since the quadratic placement is based on the squared Euclidean distance between two points, these hyperedges have to be transformed into two-point connections. This transformation is done by the net model.

For each net, the squared Euclidean distance between every two-point connection of the net is weighted and added up to the netlength. All netlengths are then added up in the cost function $\Gamma$ of the quadratic placement.

Since $\Gamma$ can be separated in x- and y-direction, just the x-direction is described further on for sake of brevity.

### 3.1. Clique Net Model

Traditionally quadratic placers utilize the clique net model for each hyperedge in the netgraph, i.e. all possible two-point connections between the modules in the net are used.

If one net has $P$ pins and the x-coordinate of each pin is denoted by $x_i$ $(i = 1, 2, 3, ..., P)$, the length of the corresponding clique net is given by

$$\Gamma_{C,x} = \frac{w_C}{2} \cdot \sum_{i=1}^{P} \sum_{j=1, j \neq i}^{P} (x_i - x_j)^2 \tag{16}$$

The net weight $w_C$ of clique model is specified by

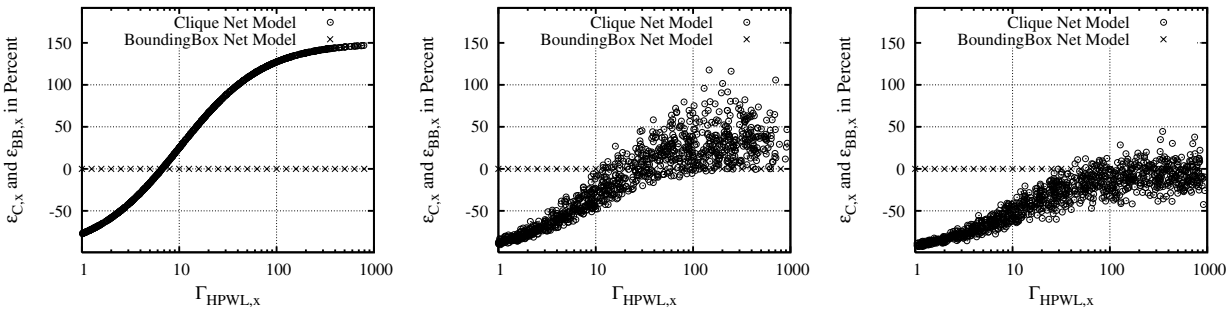
$$w_C = \frac{1}{P} \cdot \frac{2}{P} \cdot \lambda \tag{17}$$

Factor $\frac{1}{P}$ in the net weight adapts the clique model to the star model [21], [32]. Factor $\frac{2}{P}$ is to adjust the total net weight to the number of edges in a spanning tree connecting all pins of the net [19]. The additional net weight $\lambda$ can be used to linearize the quadratic clique length $\Gamma_{C,x}$ [29].

The number of connections in the clique model is determined by

$$N_{C,con} = 0.5 \cdot P(P-1) \tag{18}$$

The squared clique length (16) is one metric to measure netlength. As the placement process should consider the routing process, the netlength in the placement process should reflect the routed wirelength. Since nets are routed with horizontal and vertical wires, the minimal routed wirelength is the length of the rectilinear Steiner minimal tree (RSMT) [14]. Because the RSMT problem is NP-hard, an efficient estimation




(a) 2 Pins (b) 5 Pins (c) 20 Pins

Fig. 3. Approximation error $\epsilon_{C,x}$ and $\epsilon_{BB,x}$ between the clique net model respectively the BoundingBox net model and the HPWL metric $\Gamma_{HPWL,x}$.

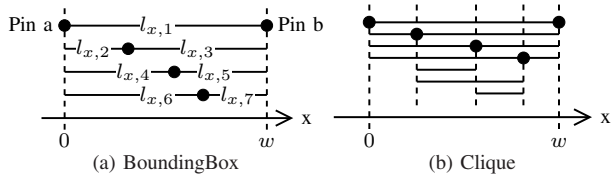
and lower bound for routed wirelength is half-perimeter wirelength (HPWL). For nets with two or three pins, the RSMT length is equal to the HPWL [14]. [10] demonstrates that for a set of circuits the HPWL differs from RSMT length by just $8\%$ but is $1.4 \times 10^5$ times faster determined. The HPWL is defined per net by the half-perimeter of the bounding box enclosing its pins. If this box has width $w$ and height $h$, then the HPWL is calculated by

$$\Gamma_{HPWL} = \Gamma_{HPWL,x} + \Gamma_{HPWL,y} = w + h \tag{19}$$

Since the HPWL is an efficient estimation for routed wirelength, the netlength in quadratic placement should reflect the HPWL.

Figures 3 (a), (b), and (c) illustrate the approximation error $\epsilon_{C,x} = \frac{\Gamma_{C,x}}{\Gamma_{HPWL,x}} - 1$ between the clique netlength and the HPWL metric of randomly generated nets. To linearize the quadratic clique length (16), the additional net weight $\lambda$ was set to $\lambda = \frac{10}{\Gamma_{HPWL,x}+10}$. The figures demonstrate that the approximation error of the clique net model is not constant over $\Gamma_{HPWL,x}$, is up to $150\%$, is spreading at nets with more than two pins, and is decreasing in proportion to the number of pins in a net. Even if the approximation error depends on the net weight $\lambda$, the above statements are valid. Therefore the clique net model approximates the HPWL metric and thus the routed wirelength very inaccurately. A new linear net model called BoundingBox net model is presented in the following that reproduces the HPWL without error.

### 3.2. BoundingBox Net Model




(a) BoundingBox (b) Clique

Fig. 4. BoundingBox and clique net model of a five pin net in x-direction.

In the BoundingBox net model, a hyperedge in the netgraph is not transformed into all possible two-point connections, as it is done in clique net model, but only in a few characteristic ones, as illustrated in figure 4(a): Pin $a$ with lowest x coordinate is connected with pin $b$ with highest x coordinate. This creates connection 1 with length $l_{x,1} = w$. The remaining $P-2$ inner pins of the net are connected with both outer pins $a$ and $b$, which creates connections $j$ and $j+1$ with $j = 2, 4, 6, ..., 2(P-2)$ and $l_{x,j} + l_{x,j+1} = w$. Considering that the pins $a$ and $b$ are the bounds of the net's box, the BoundingBox net model is characterized that all its connection are joined to the bounds of this box.

All together there are

$$N_{BB,con} = 1 + 2(P-2) \tag{20}$$

connections in the BoundingBox net model. and each connection $i = 1, 2, 3, ..., N_{BB,con}$ has the weight $w_{x,i}$. The length of every connection is squared, weighted and added to the netlength $\Gamma_{BB,x}$ of this net model:

$$\Gamma_{BB,x} = \frac{1}{2} \sum_{i=1}^{N_{BB,con}} w_i \cdot l_{x,i}^2 \tag{21}$$

With each weight calculated by

$$w_{x,i} = \frac{2}{P-1} \frac{1}{l_{x,i}} \tag{22}$$

the quadratic netlength $\Gamma_{BB,x}$ of the BoundingBox net model in x-direction is equal to the width $w$ of the net's bounding box and therefore expresses linear netlength.

Proof:

$$\begin{aligned} \Gamma_{BB,x} &= \frac{1}{2}\Big[w_{x,1} \cdot l_{x,1}^2 + \\ &\quad + \sum_{j=1}^{P-2} \left(w_{x,2j} \cdot l_{x,2j}^2 + w_{x,2j+1} \cdot l_{x,2j+1}^2\right)\Big] \\ &= \frac{1}{P-1}\Big[l_{x,1} + \sum_{j=1}^{P-2} \left(l_{x,2j} + l_{x,2j+1}\right)\Big] \\ &= \frac{1}{P-1}\big[w + (P-2) \cdot w\big] = w \end{aligned} \tag{23}$$

■

With $w = \Gamma_{HPWL,x}$ this yields an approximation error $\epsilon_{BB,x} = \frac{\Gamma_{BB,x}}{\Gamma_{HPWL,x}} - 1$ of zero in BoundingBox net model as is shown in figures 3 (a), (b) and (c).

Please note that in each iteration of the placement algorithm (see figure 1), the bounds as well as the weights $w_{x,i}$ are determined for each net.

By creating the y-part of the BoundingBox net model similarly to the above described x-part, the netlength $\Gamma_{BB} = \Gamma_{BB,x} + \Gamma_{BB,y}$ of the BoundingBox net model is equal to the HPWL $\Gamma_{HPWL} = w + h$ of the net. As the cost function $\Gamma$ of quadratic placement expresses the sum of all netlengths, this cost function is equal to the HPWL of all nets and therefore estimates the total routed wirelength efficiently by using the BoundingBox net model.

### 3.3. Advantages of the BoundingBox Net Model

Because the BoundingBox net model is based on two-point connections, it can be used in any quadratic placer. Thus all quadratic placers can now represent the HPWL, as a linear metric for netlength and an efficient estimation for routed wirelength, exactly in the quadratic objective function. An important disadvantage of quadratic placers compared to nonlinear-optimization-based placers like APlace [18] and mPL [7] has been eliminated in this way, while the CPU time advantage of quadratic placers is maintained.

(18) describes that the number of connections depends quadratically on the number of pins in the clique model. (20) shows that in the BoundingBox net model the number of connections depends only linearly on the number of pins, and thus is much smaller than in the clique net model. Therefore, the memory usage is much smaller in the BoundingBox net model than in the clique net model, since every entry in matrix $\mathbf{C_x}$ needs some memory and the number of matrix entries is proportional to the number of connections. Consequently, the CPU time is lower in the BoundingBox model than in the clique model, as the CPU time of conjugate-gradient solver used to solve (14) is smaller with smaller number of connections.

With two pins fixed and no additional forces, all remaining inner pins are located at the same position in the clique model. This is because the inner connections, existing in this net model (see figure 4(b)), pull together the inner pins. Hence the modules connected to the inner pins overlap a lot and the module overlap is hard to reduce in the clique model. The problematic inner connections do not exist in the BoundingBox net model and therefore this net model does not tend to glue the inner pins together, but supports the reduction of module overlap much better than the clique model.

## 4. Results

All presented results in this paper (except those in section 4.3) are based on circuits of the ISPD-2005 benchmark suite [23] and the following machine configuration: AMD Athlon Opteron 248, 2.2 GHz, 8 GB RAM. Please note that for a fair CPU time comparison we used only one of the two available CPU cores. The wirelength as a measure of placement quality is given as total HPWL representing the sum of all netlengths calculated by the half-perimeter wirelength.

### 4.1. Clique and BoundingBox Net Model

Table 1 shows the comparison between the traditional clique net model and our new linear BoundingBox net model. Since the number of connection is much smaller in our net model, the memory usage is about $75\%$ lower and the CPU time is $23\%$ lower using our net model.

| Circuit | Clique Net Model | | BoundingBox Net Model | |
|---|---|---|---|---|
| | HPWL [m] | CPU [s] | HPWL [m] | CPU [s] |
| adaptec2 | 102.04 | 600 | 95.91 | 800 |
| adaptec4 | 219.79 | 1080 | 202.90 | 1580 |
| bigblue1 | 109.03 | 820 | 101.19 | 930 |
| bigblue2 | 171.44 | 1350 | 157.53 | 1120 |
| bigblue3 | 384.55 | 6160 | 346.01 | 5200 |
| bigblue4 | 962.32 | 13890 | 869.75 | 8400 |
| | Average: | Total: | Average: | Total: |
| | 1.00 | 6.64h | 0.9189 | 5.05h |
| Improvement to Clique | | | $\mathbf{8.11\%}$ | $\mathbf{22.8\%}$ |

Table 1. The BoundingBox net model compared to the clique net model based on circuits of the ISPD-2005 benchmark suite.

In contrast to the clique net model, our new linear net model reflects the HPWL in the quadratic cost function exactly. Therefore, the quality of placement, measured in HPWL and thus representing an efficient estimation for routed wirelength, is about $8\%$ better in the BoundingBox net model than in the clique net model.

### 4.2. Comparison with other Placers

The comparison between our placer and other state-of-the-art placers is given by table 2, where the results of the other placers are taken from [17]. Quality of placement for each placer is described in the column "Average" being the average ratio between the placer's HPWL of all five circuits compared to APlace's HPWL. Since the authors of [17] just mention that they use a 1.6 GHz machine, we scaled their CPU time results according to the ratio between the CPU frequencies: $1.6/2.2$. Only the CPU times of APlace and Capo are given for the ISPD-2005 benchmarks in [17].

The leading APlace is on average $5.6\%$ better than our placement approach but needs $16\times$ more CPU time. Looking at single results reveals that our placer has even the best result at bigblue3 circuit. Compared with Capo our placer is $9.2\%$ better and $5.4\times$ faster.

The presented results of our placer are based on quality control parameter $\mu_T = 20$. As is shown by the trade-off figure 2, which is on the basis of the same circuits, the CPU time could be improved by $50\%$ at a quality loss of $2\%$ or the quality could be improved by $0.5\%$ at a CPU time increase of $70\%$.

Since our placement algorithm utilized a simple but fast method for final placement, we improved the quality of our placement (at $\mu_T = 20$) by using the open source RowIroning package [3], which applies a branch-and-bound placer in sliding windows. Hence the original quality was improved by about $2\%$ with a CPU time increase by $60\%$.

### 4.3. ISPD-2006 Placement Contest

In the recent ISPD-2006 placement contest [1], in which placement quality is measured by a combination of total HPWL netlength, CPU time and respecting a given module target density , our placer produces the best results. Compared to mPL[8], which offers the second best results, APlace, and Capo, our placer is $1\%$, $11\%$, and $26\%$ better respectively.

| Placer | Circuit | | | | | | Average | CPU [h] |
|---|---|---|---|---|---|---|---|---|
| | adaptec2 | adaptec4 | bigblue1 | bigblue2 | bigblue3 | bigblue4 | | |
| APlace [18], [17] | 87.31 | 187.65 | 94.64 | 143.82 | 357.89 | 833.21 | 1.000 | 82.33 |
| Our Placer (Utilizing RowIroning [3]) | 93.84 | 199.75 | 99.61 | 155.19 | 339.20 | 857.09 | 1.041 | 8.70 |
| Our Placer | 95.91 | 202.90 | 101.19 | 157.53 | 346.01 | 869.75 | 1.059 | 5.05 |
| mFAR [16] | 91.53 | 190.84 | 97.70 | 168.70 | 379.95 | 876.26 | 1.064 | n/a |
| Dragon [34], [31] | 94.72 | 200.88 | 102.39 | 159.71 | 380.45 | 903.96 | 1.083 | n/a |
| mPL [8] | 97.11 | 200.94 | 98.31 | 173.22 | 369.66 | 904.19 | 1.091 | n/a |
| FastPlace [32] | 107.86 | 204.48 | 101.56 | 169.89 | 458.49 | 889.87 | 1.155 | n/a |
| Capo [4], [28] | 99.71 | 211.25 | 108.21 | 172.30 | 382.63 | 1098.76 | 1.166 | 27.49 |
| NTUPlace [9] | 100.31 | 206.45 | 106.54 | 190.66 | 411.81 | 1154.15 | 1.206 | n/a |
| FengShui [5] | 122.99 | 337.22 | 114.57 | 285.43 | 471.15 | 1040.05 | 1.494 | n/a |

Table 2. Results of our placer compared to other state-of-the-art placers based on the ISPD-2005 benchmark suite [23].

## 5. Conclusion

Based on a novel non-heuristic force modeling, a fast and robust quadratic placer was presented in this paper. Convergence to an overlap-free placement is guaranteed and engineering change order (ECO) is efficiently supported. In addition, a quality control with just one parameter was proposed to find the trade-off between CPU time and quality.

Since the move force, which distributes the modules on the chip, is calculated by a general electrostatic potential, this potential can be extended to handle for example temperature aware or congestion driven placement.

Beside the quadratic placer, an universal linear net model was described. This linear net model expresses exactly the routed wirelength measured by HPWL in a quadratic cost function, which can be minimized quite efficiently.

## 6. Acknowledgments

The authors thank the reviewers for their useful suggestions, J. Lienig for his valuable comments and U. Schlichtmann for his continuous support.

## References

[1] International symposium on physical design. http://www.ispd.cc.

[2] International technology roadmap for semiconductors. http://public.itrs.net.

[3] Ucla/umich physical design tools. http://vlsicad.eecs.umich.edu/BK/PDtools.

[4] S. N. Adya, S. Chaturvedi, J. a Roy, D. A. Papa, and I. L. Markov. Unification of partitioning, placement and floorplanning. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 550–557, 2004.

[5] A. R. Agnihorti, S. Ono, C. Li, M. C. Yildiz, A. Khathate, C.-K. Koh, and P. H. Madden. Mixed block placement via fractional cut recursive bisection. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 24(5):748–761, May 2005.

[6] U. Brenner and M. Struzyna. Faster and better global placement by a new transportation algorithm. In *ACM/IEEE Design Automation Conference (DAC)*, pages 591–596, June 2005.

[7] T. Chan, J. Cong, and K. Sze. Multilevel generalized force-directed method for circuit placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 185–192, 2005.

[8] T. F. Chan, J. Cong, M. Romesis, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: A robust multilevel mixed-size placement engine. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 227–229, 2005.

[9] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang. NTUplace: A ratio partitioning based placement algorithm for large-scale mixed-size designs. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 236–238, 2005.

[10] C. Chu. FLUTE: Fast lookup table based wirelength estimation technique. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 696–701, 2004.

[11] W. Donath. Complexity theory and design automation. In *ACM/IEEE Design Automation Conference (DAC)*, volume 19, pages 412–419, 1980.

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

[13] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, Nov. 1970.

[14] M. Hanan. On Steiner's Problem with Rectiliner Distance. *SIAM Journal of Applied Mathemetics*, 14(2):255–265, 1966.

[15] B. Hu and M. Marek-Sadowska. FAR: Fixed-points addition & relaxation based placement. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 161–166, 2002.

[16] B. Hu and M. Marek-Sadowska. Multilevel fixed-point-addition-based vlsi placement. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 24(8):1188–1203, Aug. 2005.

[17] A. B. Kahng, S. Reda, and Q. Wang. Architecture and details of a high quality, large-scale analytical placer. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 890–897, 2005.

[18] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 24(05):734–747, May 2005.

[19] 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.

[20] M. Kowarschik and C. Weiß. DiMEPACK — A Cache-Optimized Multigrid Library. In H. Arabnia, editor, *Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA 2001)*, pages 425–430. CSREA Press, June 2001.

[21] M. C. V. Lier and R. H. J. M. Otten. Planarization by transformation. *IEEE Transactions on Circuits and Systems CAS*, 20(2):169–171, Mar. 1973.

[22] K. G. Murty and F.-T. Yu. Linear complementary, linear and nonlinear programming. http://ioe.engin.umich.edu/people/fac/books/murty/linear_complementarit%y_webbook/.

[23] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 placement contest and benchmark suite. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 216–219, May 2005.

[24] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia, and A. B. Kahng. A fast hierarchical quadratic placement algorithm. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 25(4):678–691, Apr. 2006.

[25] W. Naylor, R. Donelly, and L. Sha. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer. *U.S. Patent 6301693*, Oct. 2001.

[26] B. Obermeier and F. M. Johannes. Quadratic placement using an improved timing model. In *ACM/IEEE Design Automation Conference (DAC)*, pages 705–710, San Diego, June 2004.

[27] B. Obermeier and F. M. Johannes. Temperature-aware global placement. In *Asia and South Pacific Design Automation Conference*, volume 1, pages 143–148, Yokohama, Japan, Jan. 2004.

[28] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov. Capo: Robust and scalable open-source min-cut floorplacer. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 224–226, 2005.

[29] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: A linear or a quadratic objective function? In *ACM/IEEE Design Automation Conference (DAC)*, pages 427–432, San Francisco, 1991.

[30] W.-J. Sun and C. Sechen. Efficient and effective placement for very large circuits. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 170–177, 1993.

[31] T. Taghavi, X. Yang, and B.-K. Choi. Dragon2005: Large-scale mixed-size placement tool. In *ACM/SIGDA International Symposium on Physical Design (ISPD)*, pages 245–247, 2005.

[32] N. Viswanathan and C. C.-N. Chu. Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 24(5):722–733, May 2005.

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

[34] X. Yang, B.-K. Choi, and M. Sarrafzadeh. A standard-cell placement tool for designs with high row utilization. In *IEEE International Conference on Computer Design (ICCD)*, pages 45–47, Sept. 2002.