Optimization in VLSI Design.

Global Placement.

Jens Egeblad

September 23th, 2004
Overview

- Introduction
- Modules
- Pins
- Routing and Placement
- Net Models
- The Placement Problem
- Solving the Unconstrained Problem
- End of Part I
- Placement Methods Overview
- Graph Partitioning
- Placement Based Partitioning
- Force-Based placement
- Summary
Introduction

The objective of placement is to transform logic description:

into physical minimal wire-length layout of modules:

**Ingredients:** Chip-area, modules, pins, wires and IO-pins.  

**Note:** We simplify routing during placement and do not worry too much about wires, just yet. *We leave that to the routing problem and MZ!*

The Placement Problem is **NP-hard**!
Modules, Pins and Nets

**Chip-area** is a rectangular representation of the chip’s boundary.

**Modules** are black-box rectangles and hides the underlying logic, but may be as simple as or-gates or as complex as FPU's.

**Pins** are connection points on the modules which are used by wires. May be placed everywhere “on” the module. IO-pins are connections on the chip-area (Often called IO-pads or pads).

**Nets** are wires which connect modules. Note that one wire or net may connect more than two pins.
Problem Types

Gate-array

Module
Routing area
IO-Pad

Standard-cell

Module
Routing Channel
IO-Pad

General-Cell

Module
IO-Pad

Mixed-cell

Module
Routing Channel
IO-Pad

Note: Older articles on standard-cells often assume that pins are placed in the center of the modules. This simplifies the problem since orientation is no longer an issue.

Also: Today routing is usually done on separate chip-layers, so no need for space to wires.
How Many Modules?

This is a *real* IBM-chip.

- 54,930 modules.

And this is *not* a big circuit!
Realization of Nets

Wires are placed on separate layers today but earlier modules and wires were often placed on the same layer of the chip.

Wires are only horizontal and vertical. But there may be more angles in future which actually is/has been part of DIKUs Steiner-Tree research.
Connection Graph

Nets induce a connection graph of the circuit. The connection graph is a hypergraph.

A hypergraph $H = (V, E)$ is a graph where an edge $e$ is a subset of $V$. Ordinary graphs have $|e| = 2$ for $e \in E$.

Let $H = (V, E)$ be the connection graph. Then $V$ is the modules. $E$ is the nets.

In general 95\% of the modules are connected to less than 10 nets, so $H$ is very sparse.

In general 95\% of the nets connects less than 10 pins, so $|e|$ for $e \in E$ is relatively small.
Our objective!!!

To place modules such the total net-length is minimized.

When we know the positions of the pins, determining the minimum net-length for one net is obviously a Rectilinear Steiner-Tree Problem which is NP-hard (sigh).

So we need a different formulation.
Bounding-Box Net-Length

The Bounding Box of all pins.

We define

\[ BB(n) = h + w, \]

where \( w \) and \( h \) are the width and height of the bounding-box of the pins from \( n \).

Let \( RSMT(n) \) be the value of the Rectilinear Steiner Minimal Tree spanning the pins from net \( n \) then:

\[ RMST(n) \leq \frac{\left\lceil \sqrt{n} \right\rceil}{2} + \frac{3}{2} BB(n) \]
Quadratic Star Net-Length

Determine a point - *The Star Point*:

\[
\begin{pmatrix}
    s_x \\
    s_y
\end{pmatrix}
= \frac{1}{|n|} \sum_{p \in n} \begin{pmatrix}
    x(p) \\
    y(p)
\end{pmatrix}
\]

Then the Quadratic Star Net-Length of net \( n \) is:

\[
ST(n) = \sum_{p \in P} (x(p) - s_x)^2 + (y(p) - s_y)^2.
\]
Other Net-Models

RSMT       RMST

BB          Star

Clique

However, Bounding-Box and Star are most common!
How Many Nets?

Circuit Decoder with Star Nets

54,930 modules, 59,256 Nets, 185,905 Pins.
The Placement Problem (Formulation)

Given set of modules $M$ and nets $N$ solve:
\[
\min \sum_{n \in N} w(n)L(n)
\]

Such that no two modules $m \in M$ overlap.

Where $L(n)$ is either $RSMT(n)$, $ST(n)$ or $BB(n)$.

$w(n)$ is the weight of net $n$. E.g. some nets may have higher priority.

Note: Usually the solution of the placement-problem is divided in two parts; Global and detailed (or final) placement.

Global Placement: Determine a quick solution which is a good starting point for further work. Usually deals with many cells at one time. Time varies from minutes to few hours.

Detailed Placement: Improve a global placement solution. Usually based local search methods. Time varies from hours to days.
Star Net-Length in Matrix Form (arrghh!).

\[
\sum_{n \in \mathcal{N}} w(n) ST_{2x}(n) = \tilde{x}^t \mathbf{C} \tilde{x} + d^t \tilde{x} + f \tag{1}
\]

where \( \mathbf{C}, d \in \mathbb{R}^t \) and \( f \in \mathbb{R} \) are defined as follows:

\[
c_{ij} = \begin{cases} 
\sum_{n \in \mathcal{N}} \sum_{\substack{p \in n \in \mathcal{N} \ni c(p) \notin \mathcal{F} \atop M(c(p)) = i}} w(n) & \text{if } i = j \text{ and } i \leq s \\
\sum_{n \in \mathcal{N}} \sum_{\substack{p \in n \in \mathcal{N} \ni c(p) \notin \mathcal{F} \atop M(c(p)) = i}} -w(n) & \text{if } i = j \text{ and } i > s \\
\sum_{n \in \mathcal{N}} \sum_{\substack{p \in n \in \mathcal{N} \ni c(p) \notin \mathcal{F} \atop M(c(p)) = i}} -w(n) & \text{if } j \leq s \text{ and } i > s \\
0 & \text{otherwise}
\end{cases}
\]

\[
d_i = \begin{cases} 
2 \sum_{n \in \mathcal{N}} \sum_{\substack{p \in n \in \mathcal{N} \ni c(p) \notin \mathcal{F} \atop M(c(p)) = i}} \text{ofs}_{x}(p) & \text{for } i \leq s \\
-2 \sum_{p \in N-1(i)} \text{ofs}_{x}(p) - 2 \sum_{p \in N-1(i)} x(c(p)) & \text{for } i > s
\end{cases}
\]

\[
f = \sum_{n \in \mathcal{N}} \left( \sum_{\substack{p \in n \ni c(p) \notin \mathcal{F} \atop \mathcal{F}}} w(n)(x(c(p))^2 + 2x(c(p))\text{ofs}_{x}(p)) + \sum_{p \in n} \text{ofs}_{x}(p)^2 \right)
\]

(Homework: Prove this! - No just kidding!)

How large is the matrix \( \mathbf{C} \)? Is it sparse?
Minimization of Star Length.

The unconstrained star-length problem is:

$$\min_{\bar{x}} g(\bar{x}),$$

where $$g(\bar{x}) = \bar{x}^t C \bar{x} + d^t \bar{x} + f.$$ 

$$g(\bar{x})$$ is a quadratic function and its extreme point $$\bar{x}'$$ occurs for $$\nabla g(\bar{x}') = 0.$$ 

$$C$$ is symmetric so we get.

$$\nabla g(\bar{x}) = C^t \bar{x} + C \bar{x} + d = 2C \bar{x} + d$$

Therefore $$\bar{x}'$$ must solve:

$$2C \bar{x}' = -d$$

Which is a huge linear equation we can solve quickly with a common tool of the computer science trade:

**Conjugate Gradient Method**
Conjugate Gradient Method

When $A$ is symmetric:

$$A = A^t,$$

and positive semidefinite:

$$x^tAx > 0, \quad \text{for } x \neq 0$$

We can solve:

$$Ax = b$$

using the Conjugate Gradient Method.

Works by taking small clever steps towards the solution.

Running time is $O(\sqrt{km})$ where $k$ is the condition number of the matrix and $m$ is the number of non-zeroes in the matrix.

Only efficient when $A$ is sparse. Solving large instances with 10,000 modules requires around 50-100 steps. Each step is dominated by the running time of a sparse matrix multiplication.
BB formulation.

The *unconstrained* BB Net-Length can be solved as a linear program:

\[
\begin{align*}
\text{min} & \quad \sum_{n \in \mathcal{N}} w(n) (\overline{x_n} - x_n + \overline{y_n} - y_n) \\
\text{subject to:} & \\
\overline{x_n} & - x(c(p)) \quad \overline{y_n} \quad - y(c(p)) & \overset{\geq}{\forall} & (\text{o}fs_x(p)) & (\text{o}fs_y(p)) & n \in \mathcal{N}, p \in n, c(p) \notin \mathcal{F} \\
x(c(p)) & - x_n \quad y(c(p)) & \overset{\leq}{\forall} & \max_{p \in n}(x(c(p)) + \text{o}fs_x(p)) & \max_{p \in n}(y(c(p)) + \text{o}fs_y(p)) & n \in \mathcal{N}, c(p) \in \mathcal{F} \\
- x_n \quad - y_n & \overset{\leq}{\forall} & \min_{p \in n}(x(c(p)) + \text{o}fs_x(p)) & \min_{p \in n}(y(c(p)) + \text{o}fs_y(p)) & n \in \mathcal{N}, c(p) \in \mathcal{F}
\end{align*}
\]

Actually the unconstrained LP-formulation of BB is the dual of a network flow problem and can be solved more efficiently than standard LPs. However the running time is \(O(mn \log(n^2/m))\) where \(m = |\text{modules}| + |\text{nets}|\), and \(n = |\text{nets}| + |\text{pins}|\).
The Unconstrained Solutions

Three circuits; macro, standard-cell and mixed.

Conclusion: Completely unconstrained BB formulation generates useless solutions.
End of Part I

Summary so far:

- **Ingredients:** modules, nets, pins.
- **Connection graph** Nets induce a hypergraph.
- **Problems are huge:** Many modules, nets and pins!
- **Routing:** Optimal is RSMT but we use various net-models.
- **Main net-models:** Star and BB.
- **Unconstrained Problem:** Both star and BB formulation.
- **Unconstrained Star:** Is quadratic function with symmetric positive semidefinite matrix. Use Conjugate Gradient to solve. Solution contains lots of overlap but maybe useful.
- **Unconstrained BB:** Is LP. Dualize and solve net-flow problem. Solution contains excessive overlap and is completely useless.
Part II - Global Placement Techniques

We will consider three main ideas.

- Graph Partitioning.
- Geometric Partitioning.
- Force-Based Placement.

Most placement techniques use either of these three!
Graph Partitioning

Exploits the fact that “Nets induce a hypergraph!”.

The graph is as follows:

- Each module corresponds to a vertex \( v \in V \).
- Each net corresponds to an edge \( e \in E \).
- All vertices and edges in \( H = (V, E) \) have weights.
- The weight \( w(v) \) of a vertex \( v \in V \) is the area of the corresponding module.
- The weight \( w(e) \) of a edge \( e \in E \) is the “priority” of the net.
Graph Partitioning - Formulation

Somehow we wish to recursively divide the graph in two almost equal parts (with respect to area) such that the netlength is minimized.

The thought: The netlength is reduced by placing connected modules in the same partition.

Let \( \text{cut}(V_1, V_2) \) be defined as:

\[
\text{cut}(V_1, V_2) = \{ e \in E | e \cap V_1 \neq \emptyset \text{ and } e \cap V_2 \neq \emptyset \} 
\]

So we wish to find \( V_1 \) and \( V_2 \) such that:

Minimize \[ \sum_{e \in \text{cut}(V_1, V_2)} w(e). \]

Such that:

\[
\begin{align*}
    w(V_1) & \leq \alpha w(V), \\
    w(V_2) & \leq \alpha w(V),
\end{align*}
\]

where \( \frac{1}{2} \leq \alpha \leq 1. \) (Note: \( w(V') = \sum_{v \in V'} w(v) \)).
Graph Partitioning - Solving

Graph bipartitioning is NP-hard.

The most famous/successful heuristic is by Fiduccia and Mattheyses.

It works from an initial solution and moves one vertex at a time from $V_1$ to $V_2$ or vice versa.

In each step the vertex resulting in the least price for the cut is moved.

When a vertex has been moved we lock it for subsequent moves.

When all vertices have been moved and locked, we restart from the best solution encountered so far.
Graph Partitioning - Creating a Placement

Given some placement region we bisect the graph and assign modules from each partition to half of a placement region, recursively.
GORDIAN - Overview

GORDIAN (Global optimization and rectangle dissection).

The method roughly proceeds as follows:

1. Minimize the quadratic net-length of the modules.
2. Bisect the resulting placement such that the area of the modules on either side is almost equal and the number of cuts is minimized.
3. Assign each module to the appropriate region.
4. Create constraints for the modules such that they will lie within their assigned region.
GORDIAN - Constraints

The quadratic function is solved as unconstrained star netlength. However it is reformed without star-point (net-center) coordinates.

Let:

- $M_p \subseteq M$ be the modules assigned to the region $p$.
- $u_p = (u_x, u_y)$ be the center of $p$.
- $F_m$ be the area of module $m \in M$.
- $(x_m, y_m)$ be the coordinates of $m \in M$.

Then we add the constraint that the center of gravity of $M_p$ is $u_p$:

$$\frac{\sum_{m \in M_p} F_m \cdot x_m}{\sum_{m \in M_p} F_m} = u_x,$$
$$\frac{\sum_{m \in M_p} F_m \cdot y_m}{\sum_{m \in M_p} F_m} = u_y,$$

At step $j$ this corresponds to constraints in matrix form:

$$A_j \tilde{x} = u_j$$

(Each region corresponds to two rows in $A_j$ and $u_j$).
GORDIAN - Solving with Constraints (aarghh)

In step $j$ we have $q \leq 2^j$ regions and solve the constrained quadratic program:

$$\min_{\tilde{x} \in \mathbb{R}^2 |M|} \left\{ \tilde{x}^t C \tilde{x} + d^t \tilde{x} + f \mid A_j \tilde{x} = u_j \right\},$$

where $\tilde{x}$ contains $x$- and $y$-coordinates for all modules $M$.

However, since $q < |M|$, $\tilde{x}$ can be partitioned into $2 \cdot q$ dependent ($\tilde{x}_d$) and $2 \cdot (|M| - q)$ independent ($\tilde{x}_j$) variables, and thus the matrix $A_j$ can be partitioned:

$$A_j = [D_j B_j],$$

and let

$$\tilde{x}_d = -D_j^{-1} B_j \tilde{x}_i + D_j^{-1} u_j \quad \text{then} \quad \tilde{x} = Z_j \tilde{x}_i + \tilde{x}_0,$$

where

$$Z_j = \begin{bmatrix} -D_j^{-1} B_j \\ 1 \end{bmatrix} \quad \text{and} \quad \tilde{x}_0 = \begin{bmatrix} D_j^{-1} u_j \\ 0 \end{bmatrix}.$$

So instead we can solve the unconstrained problem (with CG):

$$\min_{\tilde{x} \in \mathbb{R}^2 (|M| - q)} \left\{ \tilde{x}_i^t Z_j^t C Z_j \tilde{x} + (C \tilde{x}_0 + d^t Z_j) x_i + f \right\},$$

(Read the article for details)
GORDIAN - Partitioning

How do we partition a region $p$?

GORDIAN uses the placement solution from the quadratic problem and looks for a partition with the following two properties:

1. The area of modules on either side should be roughly equal.
2. Number nets crossing the partition should be minimal.

Assume that we wish to partition $M_p$ in left and right subsets $M_l$ and $M_r$. The partition can be evaluated by:

$$\alpha = \frac{\sum_{m \in M_l} F_m}{\sum_{m \in M_r} F_m}.$$

Property 1 looks for a partition with $\alpha = \frac{1}{2}$.

Property 2 looks for a partition with an $\alpha$ such that:

$$c_p(\alpha) = \sum_{n \in N_l \cap N_r} w(n).$$

**Simple Vertical cut scheme:** Sort modules from left to right and find the minimum $c_p(\alpha)$ with $0.35 \leq \alpha \leq 0.65$. 

28
GORDIAN - Partitioning (2)

Also to further reduce $c_p(\alpha)$ GORDIAN uses Fiduccia Mattheyses min-cut algorithm to interchange modules.

Finally, if after partitioning and optimization of the new problem some of the modules are located outside their assigned region, repartition the problem!
GORDIAN - Summary

Recursively:

- Partition each region by either a vertical or horizontal cut-line.
- Interchange the modules to minimize the number of nets cut by the partition.
- Formulate the “Center-of-gravity-constraints”.
- Combine the “Center-of-gravity-constraints” with quadratic optimization and solve it as unconstrained quadratic problem with the Conjugate Gradient Method.
- Use the solution as a placement of the modules.
- While modules fall outside their region repartition and reoptimize.

Questions:

- When do you stop the recursion?
- How do you get a legal non-overlapping solution?
**Kraftwerk** - (“*She’s a module and she’s looking good*”)

**Observation:** Nets can be interpreted as springs “pulling” modules towards each other. The minimal star-netlength occurs at force equilibrium when:

\[ 2C\ddot{x} + d = 0 \]

**Kraftwerk Idea:** Add additional forces to this spring system such that modules are spread on the placement area. Solve:

\[ 2C\ddot{x} + d + e = 0, \]

where \( e \) represents the additional forces.

31
Kraftwerk - Requirements to Additional Forces

Requirements:

1. The additional force on a module depends *only* on its position.
2. Regions with many modules acts like sources.
3. Regions with few modules acts like sinks.
4. Forces do not form circles.
5. In infinity the force is 0.
Kraftwerk - Defining Additional Forces

Define:

\[ a_m(x, y) = \begin{cases} 
1 & \text{if } (x, y) \text{ covered by module } m \\
0 & \text{otherwise} 
\end{cases} \]

and

\[ A(x, y) = \begin{cases} 
1 & \text{if } (x, y) \text{ within placement area} \\
0 & \text{otherwise} 
\end{cases} . \]

Now let:

\[ D(x, y) = \sum_{m \in M} a_m(x, y) - \frac{\sum_{m \in M} F_m}{W \cdot H} \cdot A(x, y), \]

where \( W \) and \( H \) is the width and height of the placement area.

Then set the additional force at position \((x, y)\) to:

\[ f(x, y) = \frac{k}{2\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} D(x', y') \frac{r - r'}{|r - r'|^2} dx' dy', \]

where \( r = (x, y) \) and \( r' = (x', y') \)

This force obeys our requirements.
Kraftwerk - Using Additional Forces

To calculate the additional force create an \( m \times m \) bin structure on the placement area:

\[
f_{ij} = \frac{k}{2\pi} \sum_{i' = 1}^{m} \sum_{j' = 1}^{m} D_{i'j'} \frac{r_{ij} - r_{i'j'}}{|r_{ij} - r_{i'j'}|^2} w \cdot h,
\]

where \( w \) and \( h \) is the dimension of each bin, \( D_{ij} \) is the density of bin \((i, j)\) and \( r_{ij} \) is the center of bin \((i, j)\).

\( f_{ij} \) can be calculated in \( O(m^2 \log m) \) time using a Fast Fourier Transform when \( m \) is the number of bins. (Homework)
Kraftwerk - Solution Process

Kraftwerk iteratively improves the solution by solving:

\[ 2C\ddot{x} + d + e_i = 0, \]

and calculating the additional forces \( e_i \).
Kraftwerk in Action
Circuit: Primary2 (2907 modules).

(1 iter.)

(3 iter.)

(5 iter.)

(7 iter.)

(15 iter.)

(25 iter.)

(30 iter.)

(40 iter. (77 sec.))

(This is my implementation not Kraftwerk)
End of Part II - Summary

- **Ingredients**: modules, nets, pins.
- **Problems are huge**: Many modules, nets and pins!
- **Routing**: Optimal is RSMT but we use various net-models.
- **Main net-models**: Star and BB.
- **Unconstrained Star**: Is quadratic function with symmetric positive semidefinite matrix. Use Conjugate Gradient to solve. Immediate solution contains lots of overlap.
- **Graph Partitioning**: Recursively cuts minimal number of nets in connection graph with. Achieve placement by slicing structure.
- **GORDIAN**: Recursively bisect the placement area. Constrain modules to the region they fall into. Make constraint part of objective function. Improve solution by module-interchanging and repartitioning.
- **Kraftwerk**: Iteratively add additional forces to spread modules until they no longer overlap (very much).