Implementation and Analysis of Placement Algorithms based on Mechanical Methods (Force Directed) for Microelectronic Circuits

Giaourtas P. Michail

Diploma Thesis

University of Thessaly

Department of Electrical and Computer Engineering

Supervisors:

Georgios Stamoulis, Professor

Nestoras Eumorfopoulos, Assistant Professor

Volos, Greece
March 2017

Implementation and Analysis of Placement Algorithms based on Mechanical Methods (Force Directed) for Microelectronic Circuits

Υλοποίηση και Ανάλυση Αλγορίθμων Τοποθέτησης βασισμένοι σε Μεθόδους Μηχανικής (Force Directed) για Μικροηλεκτρονικά Κυκλώματα

Giaourtas P. Michail

Abstract

The construction and optimization of Integrated Circuits (ICs) is, for decades now, one of the most important parts of the electronics industry. Physical design and, specifically, placement is an essential part in the manufacturing process that ensures the proper alignment of all circuit components on a chip and, in that way, it affects its size, performance and consumption.

Within this thesis, the construction of a placement algorithm is completed which belongs to the Force-directed category. Based on the Kraftwerk2 algorithm, it simulates spring forces on target points and, using an electrostatic potential generated by a supply and demand system, it places the circuit components, aiming the least overlap with the smallest connection distances (wirelength). The analysis, discretisation and construction of the system, the calculation of the electrostatic potential, the calculation and normalization of the target points, the implementation of a new quality control factor of the final result and a new termination condition are the main experimentation parts that will lead to the completion of an algorithm with several degrees of freedom capable of producing quality results in a very short time.

Περίληψη

Η κατασκευή και βελτιστοποίηση των ολοκληρωμένων κυκλωμάτων (Integrated Circuits - ICs) αποτελεί εδώ και δεκαετίες ένα από τα σημαντικότερο κομμάτια στη βιομηχανία των ηλεκτρονικών συσκευών. Η φυσική σχεδίαση και, συγκεκριμένα, η τοποθέτηση είναι ένα απαραίτητο τμήμα στην κατασκευαστική διαδικασία που φροντίζει για τη σωστή διάταξη όλων των κυκλωματικών στοιχείων σε ένα τσιπ και με τον τρόπο αυτό επηρεάζει το μέγεθος, την απόδοση και την κατανάλωσή του.

Στα πλαίσια αυτής της διπλωματικής εργασίας ολοκληρώνεται η κατασκευή ενός αλγορίθμου τοποθέτησης που ανήκει στην κατηγορία Force-directed. Βασισμένος στον αλγόριθμο Kraftwerk2, προσομοιώνει δυνάμεις ελατηρίου πάνω σε σημεία-στόχους και, με τη χρήση ενός ηλεκτροστατικού δυναμικού παραγόμενο από ένα σύστημα προσφοράς και ζήτησης, τοποθετεί τα κυκλωματικά στοιχεία στοχεύοντας στη μικρότερη επικάλυψη (overlap) με τη μικρότερη απόσταση μεταξύ τους (wirelength). Η ανάλυση, διακριτοποίηση και κατασκευή του συστήματος, ο υπολογισμός του ηλεκτροστατικού δυναμικού, ο υπολογισμός και η κανονικοποίηση των σημείων-στόχων, η υλοποίηση ενός νέου παράγοντα ελέγχου ποιότητας του τελικού αποτελέσματος και μια νέα συνθήκη τερματισμού είναι τα βασικά σημεία πειραματισμού που θα οδηγήσουν στην ολοκλήρωση ενός αλγορίθμου με αρκετούς βαθμούς ελευθερίας ικανό να παράξει ποιοτικό αποτέλεσμα σε πολύ λίγο χρόνο.

Declaration

The work in this thesis is based on research carried out at the Department of Electrical and Computer Engineering, University of Thessaly, Greece. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I would like to thank my supervisors Dr. Christos Sotiriou, Dr. Georgios Stamoulis and Dr. Nestoras Eumorfopoulos for their guidance and constant help during the development of this thesis as well as for their patience and trust in me.

Special thanks to my colleagues Xanthos Vlachos, Nikolaos Sketopoulos, Stavros Ntentos and Angelina Delakoura for their help and cooperation during these months.

I would also like to express my gratitude to my family for their unfailing support all these years. Finally, I would like to thank my friends for their encouragement and continuous support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decades the integrated circuits industry has rapidly evolved. This is due to the continuous and constantly growing demand for electronic systems. While their complexity gets higher, integrated circuits are getting smaller and at the same time more powerful, creating a wide range of products from personal computers and portable devices to supercomputers used for scientific and engineering applications. Nowadays, the electronic devices are not only important but necessary for our communication, information research and entertainment. They also play a critical role in the field of scientific research providing tools used for experimental purposes.

The integrated circuits (ICs) are the most vital ingredient of the electronic devices. After the invention of the transistor (Bell Telephone Laboratories - 1947), the first working IC was demonstrated in 1958 from Jack Kilby at Texas Instruments, Inc.. Since the first generation, their density in transistors keeps rising till today.

In 1965, Gordon Moore, the co-founder of Intel and Fairchild Semiconductor, after observation, predicted that the number of transistors that can be integrated in a single silicon chip will increase exponentially over time (Figure 1.1). This prediction has successfully been proven for decades and know as Moore's law.

The implementation strategies and methods of the ICs have changed from time to time due to this advancement. After the circuit design stage, a number of steps have to be completed before a chip is ready. The first microprocessor, Intel 4004, was designed completely by hand, by placing each transistor on it individually, which is impossible for today's multimillion-transistor chips. Modern IC design includes an

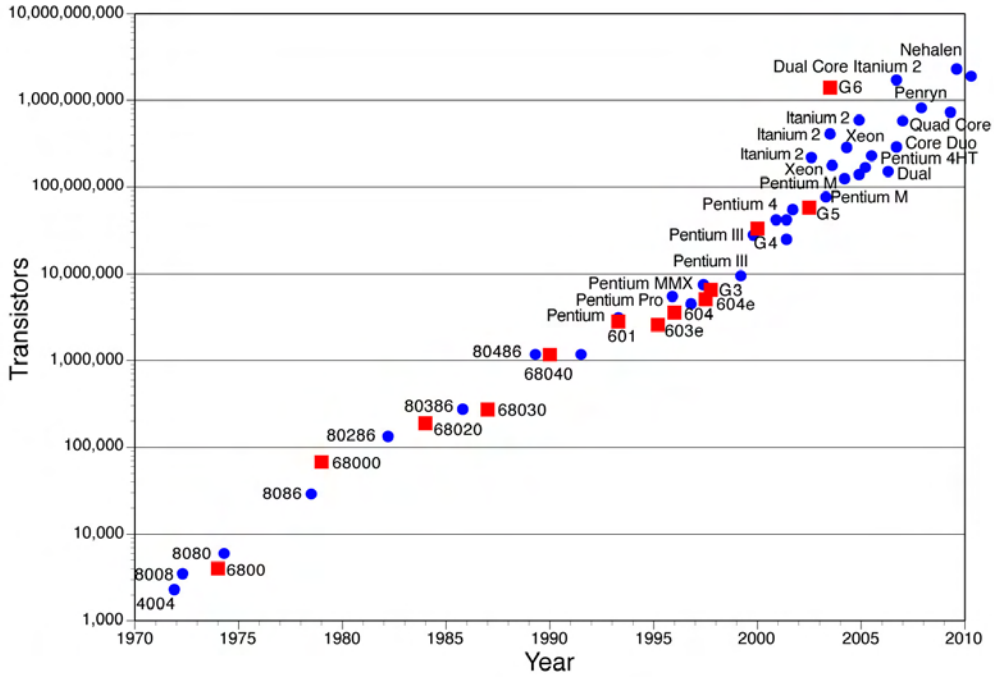

Figure 1.1: Moore's Law: Increation of the number of transistors per chip through the years till 2010.

essential step called physical design, in order to compute the near-optimum physical layout of millions to billions of circuit components on a tiny silicon surface, before the manufacturing process in the Wafer Fabrication Houses. In this process, specialized EDA tools have been developed to support and optimise the IC designs.

1.1 Physical Design and EDA Tools

With the rapid technological evolution, the launching time of a product on the market became critical. For that reason, design techniques became more and more methodical so that they could later be automated. This resulted a more subtractive way of designing, making the construction of a circuit hierarchical. In each stage, the produced design can be enclosed in a black box, keeping only the important information about it and allowing the discretization of the production processes. This divide and conquer logic accelerated the integration time of a chip, since every construction stage can be isolated and optimized, interfering as minimum as possible with the remaining stages.

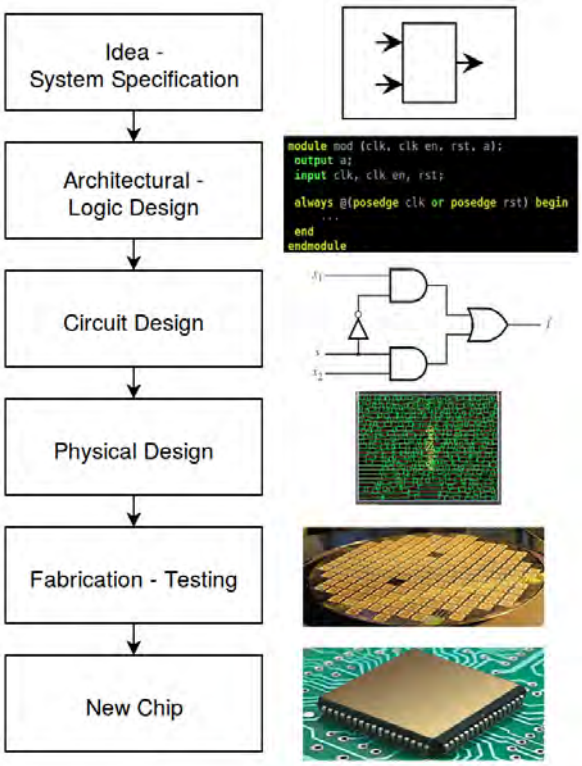

Figure 1.2: Integrated circuit design flow.

The Integrated Circuit design can be separated into tree discrete areas: the Front-end design using Hardware Description Languages (HDLs), the Verification and the Back-end design or Physical design. **Physical design** is one of the most essential areas of the IC design. Its' target is to convert the abstract circuit description that is created from the Front-end design via Hardware Description Languages (HDLs) into a 2D (or 3D) physical geometric layout, meaning an analytical map that describes the size and position of millions of circuit components and their connectivity on a small silicon surface. The final product of this phase is the input of the fabrication phase and affects the performance, the area, the power consumption and the reliability of the circuit. Physical design targets to compute the optimum arrangement of the components in order to achieve efficient interconnection schemes and, therefore, area, cost and power consumption minimization and obtain the desired functionality and performance.

The Physical design flow can be separated into 6 main steps: Floorplanning, Partitioning, Placement, Clock Tree Synthesis, Signal Routing and Timing Closure.

The first tree steps aim to manually place specific components, split the layout into discrete areas and finally place all the circuit components on the layout, optimally. The rest of the steps handle the connectivity issues that arise and ensure the functionality of the final product.

Placement is one of the most critical steps of the Physical design, since it usually handles the majority of the circuit components that will be placed on a layout. This means that a placer must be capable of placing millions of components on the layout, which inevitably has to be handled algorithmically.

Since Physical design usually handles simply shaped components, like rectangles and lines, it may be considered as a simple graphs problem. However this approach is not enough because a lot of other factors influence the result, such as the electric signals and the construction limitations. In general, Physical Design is an algorithmic problem which can be handled with Electronic Design Automation (EDA) tools.

The EDA tools are specialized software tools which implement algorithms including all the necessary limitations. In the Physical design field, EDA tools target to create better solutions in less time and they must be frequently updated to anticipate the technological developments.

1.2 Introduction to Placement

Placement is the phase of the Physical design where the basic structure of a circuit is created. It determines the exact final location of each component by eliminating the overlap between them and optimizing a cost function which usually represents the total wirelength. Interconnect delays influence chips' performance more than ever, since their density, size and performance are increasing, especially with aggressive technology scaling into the deep submicron (DSM) era. The fact that Placement sets the location of the circuit components and, hence, the overall interconnection distances, it has significant impact on the final performance of the design.

Although Placement seems to be an easy problem to solve, it has been proven that it is *NP-complete*. [1] This means that it cannot be solved in polynomial run

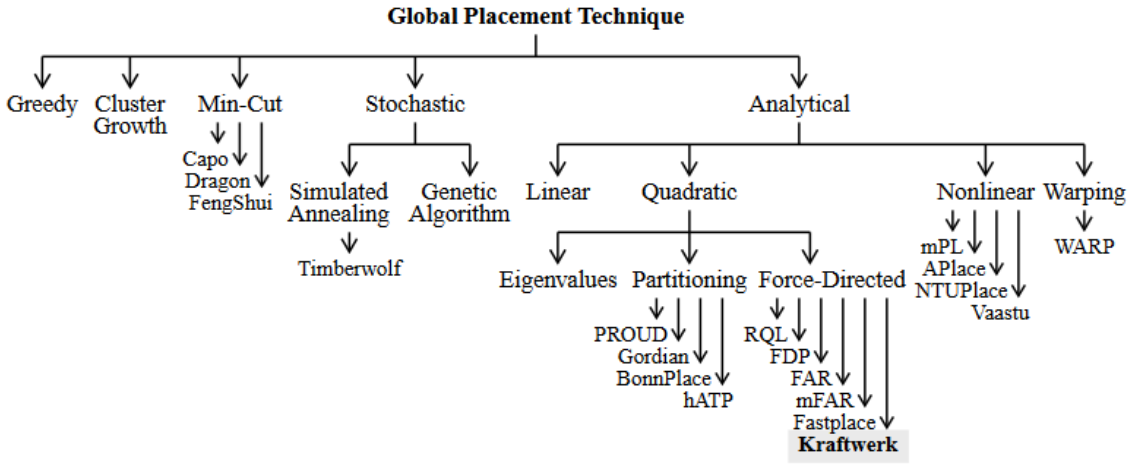

Figure 1.3: Different placement techniques and names of various placers.

time in any known way, hence, approximation solutions have to be found using heuristics.

Placement can be separated into three discrete sub problems: Global placement, Legalization, and Detailed placement. Global placement calculates and assigns the first and approximated location coordinates of each component on the chip. This initial placement may be near-optimum, usually in terms of minimizing the interconnection distances of all the components and simultaneously spreading them among the layout, but it allows them to overlap to some degree.

Legalization is the part of the placement process where all the overlapping areas are eliminated, affecting the quality of the Global placement's result as less as possible. Finally, Detailed placement improves the overall quality of the circuit by making corrections on the components' positions.

As shown in Figure 1.3, many global placement techniques have been developed thought the years. Greedy and Cluster-Growth are two bottom-up approaches which offer fast results and are mostly used for small circuits or local optimizations but can get stuck in local minimums. Min-Cut placers adapt top-down techniques using graph partitioning algorithms which are usually easy to implement but not providing the best solution in terms of placement quality. Stochastic methods choose the best result after a series of placement tests and iterations. They are adapting and can escape from local minimums but a lot of CPU demanding.

Finally, Analytical placers minimize a continuous and analytical cost function, which usually represents the wirelength, using numerical optimization methods. The cost function can either be Linear, Nonlinear (and non-quadratic) using a log-sum-exp function to express the wirelength, Warping using approaches of computational geometry or Quadratic. Specifically, Quadratic placers need an additional factor to reduce components' overlap because they use the quadratic cost function to only minimize the wirelength. Depending on that factor, they can be subdivided into three categories: based on eigenvalues, partitioning, and force-directed.

1.3 Purpose of this Thesis

This thesis focuses on a Force-directed Global placer named Kraftwerk2. As an Analytical and Quadratic placer, Kraftwerk2 offers a very qualitative result, quite fast and efficiently using a new net model and, as a Force-directed approach, it innovates by implementing two different additional forces, instead of one, using a Poisson potential. In addition, extra features like the quality control and an advanced module demand method, leave enough space for further investigation of new enhancements. Hence, the construction of Kraftwerk2 will be set as a reference point. At first, the algorithm must be studied and analyzed in order to find ways of implementing its original form, facing every development issue that may occur by researching construction solutions and replacements.

As a second step, the created algorithm will be tested using real industrial and academic benchmarks in order to observe the pros and cons of it.

Finally, new heuristics will be put to test. The main goal of this thesis is understand in depth the placer, decode the exact effect of each part, simplify it and then enhance it or exclude changes that didn't lead to better solutions.

Chapter 2

Background

2.1 Force-directed Placement

Force-directed placement techniques are known for offering high quality results in a very short time. Since they belong to the quadratic family placers (QP), they use a quadratic cost function, which is easily minimized. To create this function, a connectivity model of all components has to be set, hence the net model. Each component represents a standard cell or a macro and will be called "cell" in this thesis. A set of cells connected between them is called a net.

Due to the huge number of these connections, the analytical calculation of their routed length would be an extremely time-consuming process. Half-Perimeter Wire Length (HPWL) is a good estimation of the routed wirelength which is defined as the width and height of the smallest rectangle that encloses all the cells of a net. However, HPWL is not a quadratic function and cannot be minimized efficiently.

The sum of all nets must be represented by a net model in a useful way in order to calculate the total wirelength cost in a quadratic function, since only pair connections can be used. Mostly used quadratic net models are clique and star model which simulate each net as the sum of all possible connections between every pair of cells in the said net, or as the sum of all connections between every cell and a pin, accordingly. For example, the clique net model uses the following cost function for x and y coordinates and for each net n:

$$\Gamma_{n,x} = \frac{1}{2}\sum_{p=1}^{P}\sum_{q=p+1}^{P} w_{x,pq}(x_p - x_q)^2, \tag{2.1.1}$$

$$\Gamma_{n,y} = \frac{1}{2}\sum_{p=1}^{P}\sum_{q=p+1}^{P} w_{y,pq}(y_p - y_q)^2, \tag{2.1.2}$$

where P is the number of the cells in the net, p and q are two different cells with x_p, y_p and x_q, y_q coordinates and $w_{x,pq}$ the weight of the connection of p and q. The sum of all nets' wirelength represents the total quadratic cost:

$$\Gamma = \sum_{n \in N} \Gamma_{n,x} + \Gamma_{n,y}, \tag{2.1.3}$$

where N is the number of nets.

Quadratic placers usually complete a separate Quadratic placement step to reach the wirelength cost minimum and then proceed to a spreading procedure. Force-directed placers complete this procedure by simulating one or more additional forces in order to spread the cells and reduce their overlap. This force is usually set as an elasting spring and the overall algorithm is terminated when force equilibrium is achieved.

Many Force-directed placers have developed different approaches and techniques in order to achieve a better result. Eisenmann's Kraftwerk [2] uses Poisson's equation to create a constant force via an electrostatic potential. FAR and mFAR utilize a pair of forced to spread the components while FastPlace and RQL use fixed points to pull each component simulating a spring force.

Kraftwerk2 is a force-directed placer which combines these and other similar approaches and creates new enhancements, making it a very fast algorithm. The next section analyzes each part of the placer Kraftwerk2.

2.2 Kraftwerk2

Kraftwerk2 is an iterative, Analytical algorithm, Quadratic and Force-directed, as explained previously. This means that it is based on an additional force that is

applied to a (usually big) number of movable circuit components (cells) to eventually achieve force equilibrium and finally create a satisfying placement solution.

Kraftwerk2 innovates, separating that force in two discrete forces: the Move and the Hold force. By creating a target point for each cell, the Move force pulls them to the said points, simulating the force of an elastic spring connected between every cell and it's point. The position of each point is calculated from the gradient of an electrostatic field created by Poisson's equation, given the current cell's position.

The Hold force acts as an additional force to prevent moved cells from collapsing back to their previous position. It's value depends on the Net force that is be explained subsequently.

As a second concept of Kraftwerk2, the Bound2Bound approach is used as the net model of the algorithm. This net model calculates the weight of each two-pin connection that will later be used to determine the total wirelength cost and the Net force. The Net force acts as an elastic spring between pairs of cells and the I/O pins, pulling them back to the positions with the minimum wirelength.

This diploma thesis excludes the Bound2Bound net model approach, replacing it with the simplest Clique net model and focusing on the rest of the algorithm's parts instead. Nonetheless, each part of the initial Kraftwerk2 approach that has been mentioned is explained below, including the Bound2Bound model, as well as a number of additional features that complete the placer: the Cells' Weight, the Scale Factor k, the Halos Prevention, the Framed Supply and the Convergence.

2.2.1 Net Model and Net force

As explained, Kraftwerk2 introduces the Bound2Bound net model. This model uses the same quadratic cost function as the Clique model but with a different weight function:

$$w_{n,y}^{B2B} = \begin{cases} 0, & \text{if cell p and q are inner cells} \\ \frac{2}{P-1}\frac{1}{x_p - x_q}, & \text{else} \end{cases} \tag{2.2.4}$$

for x coordinate and respectively for y. This model ignores the inner connections of a net and takes into account only the boundary ones. This way it represents

the HPWL approach quite accurately and hence, it estimates the routed wirelength efficiently.

Since the cost function is created, it can be transformed into a matrix-vector notation:

$$\Gamma_x = \frac{1}{2}x^T C_x x + x^T d_x + const, \tag{2.2.5}$$

$$\Gamma_y = \frac{1}{2}y^T C_y y + y^T d_y + const, \tag{2.2.6}$$

where x and y are vectors which contain the position coordinates of each cell, derivatized and C_y are matrices which represent the connectivity between cells and d_x, d_y reflect the connections between cells and pins.

By derivatizing Γ for x and y and setting the derivative to zero, the minimum can be obtained.

$$\nabla_x \Gamma = C_x x + d_x = 0, \tag{2.2.7}$$

$$\nabla_y \Gamma = C_y y + d_y = 0, \tag{2.2.8}$$

This means that when the derivative is zero, the position vectors x and y contain the coordinates of the cells with the minimum wirelength cost.

In Quadratic placement, the cost function can also represent the total energy of a springs system. Hence, it's derivative can be viewed as the sum of elastic forces between cells, meaning the Net force F^{net}:

$$F_x^{net} = C_x x + d_x, \tag{2.2.9}$$

$$F_y^{net} = C_y y + d_y. \tag{2.2.10}$$

However, Net force is not enough. To spread cells in order to eliminate the overlap, Kraftwerk2 introduces a set of two additional forced: the Hold force and the Move force.

2.2.2 Hold force

Kraftwerk2 is an iterative algorithm. To prevent cells from collapsing back to the positions of the previous iteration, an additional force is added except the Net force and the one that will spread the cells (Move force): the Hold force. Hold force equals to the Net force of the previous iteration but has an opposite direction.

$$F_x^{hold} = -(C_x x' + d_x), \tag{2.2.11}$$

$$F_y^{hold} = -(C_y y' + d_y), \tag{2.2.12}$$

where x' and y' the position vertexes at the start of the iteration. This way, at the start of each iteration and before the new Move force is applied, there is a force equilibrium between cells.

2.2.3 Move force

One of the most important concepts of Kraftwerk2 is the construction of the Move force. At first, depending on the current cells position, a Demand and Supply System is created:

$$D(x, y) = D_{dem}(x, y) - D_{sup}(x, y), \tag{2.2.13}$$

where $D_{dem}(x, y)$ is the demand of cells and $D_{sup}(x, y)$ is the supply of the placement area for x and y. More specifically,

$$D_{dem}(x, y) = \sum_{i=1}^{n} d_i * R(x, y; x_i' - \frac{w_i}{2}, y_i' - \frac{h_i}{2}, w_i, h_i), \tag{2.2.14}$$

where d_i is the density of each cell i (can also be set to one) and R() is a rectangle function which returns one at the points inside the cell, or zero otherwise:

$$R(x, y; x_{ll}, y_{ll}, w, h) = \begin{cases} 1, & \text{if } 0 \leq x - x_{ll} \leq w \wedge 0 \leq y - y_{ll} \leq h \\ 0, & \text{else} \end{cases} \tag{2.2.15}$$

where x, y the input point coordinates, x_{ll}, y_{ll} the cell's lower left corner coordinates and w, h the cell's width and height. Also,

$$D_{sup}(x,y) = d_{sup} * R(x,y; x_{chip}, y_{chip}, w_{chip}, h_{chip}), \tag{2.2.16}$$

where $x_{chip}, y_{chip}, w_{chip}$ *and* h_{chip} are the placement area's constrains and d_{sup} the module supply density, set as:

$$d_{sup} = \sum_{i=1}^{n} \frac{d_i * A_i}{A_{chip}}, \tag{2.2.17}$$

where A_i the cell's i area and A_{chip} the total placement's area.

Since the construction of the system is complete, it can be used to create the electrostatic potential Φ:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \Phi(x,y) = -D(x,y). \tag{2.2.18}$$

Using the partial derivative of the potential function for each point where the cells are placed, the gradient is calculated which will lead to the creation of target points:

$$\mathring{x}_i = x'_i - \frac{\partial}{\partial x}\Phi(x,y)\Big|_{(x'_i,y'_i)} \tag{2.2.19}$$

$$\mathring{y}_i = y'_i - \frac{\partial}{\partial y}\Phi(x,y)\Big|_{(x'_i,y'_i)} \tag{2.2.20}$$

This way, instead of moving cells directly to these points, Kraftwerk2 creates target points and simulates forces that pull cells to them like a spring. According to Hook's law, the Move force can be set as

$$F^{move} = F_x^{move} + F_y^{move} = \mathring{C}_x * (x - \mathring{x}) + \mathring{C}_y * (y - \mathring{y}). \tag{2.2.21}$$

$\mathring{C}_x, \mathring{C}_y$ are diagonal matrices filled with the weight constrain $\mathring{w}_i$ which represents the spring constant of the force. The values of $\mathring{w}_i$ will be explained below.

2.2.4 Quality Control

Another important feature of Kraftwerk2 is the ability to control the overall quality and execution time. This can be achieved by adjusting the weight factor $\mathring{w}_i$ of the Move force. At first, it is initialized as

$$\mathring{w}_i = \frac{A_i}{A_{avg}} * \frac{1}{n}, \tag{2.2.22}$$

where A_{avg} is the average cell area and n the number of cells. Since Move force is a simulation of an elastic spring force, by adjusting the spring constant, hence the $\mathring{w}_i$, Move force can increase or decrease. To do that, a new factor is applied to $\mathring{w}_i$:

$$\mathring{w}'_i = \mathring{w}_i * k(\mu). \tag{2.2.23}$$

$k(\mu)$ is calculated from the function $\mu = 1 + \tanh\left(\ln\left(\frac{\mu_T}{\mu}\right)\right)$, where μ is the average cell movement in one iteration and μ_T is the user-set target average movement. As a result, the user can set a high μ_T for a fast placement result or low for better placement quality.

2.2.5 Halos Prevention

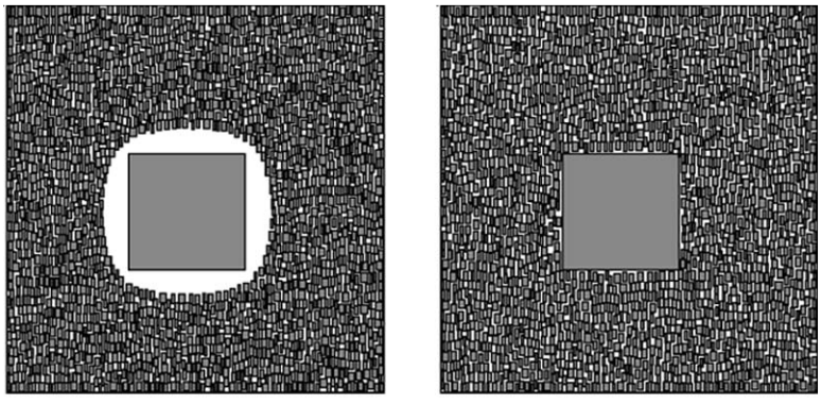

Figure 2.1: a) Halo around the large cell. b) Halo elimination with cells' density scale down.

Figure 2.1 shows the a phenomena that can occur when large cells are placed. These halos are empty regions created around them and can be corrected by scaling down the cells' density, hence the d_i factor:

$$d_i = \begin{cases} 1, & \text{if } A_i < A_{large} \\ \sqrt{\frac{A_{large}}{A_i}}\,(1\text{–}d_{sup}) + d_{sup}, & \text{else} \end{cases} \tag{2.2.24}$$

A_{large} is usually 50 times the average cell area.

2.2.6 Convergence

Since the algorithm aims to normalize the electrostatic potential in each iteration, Kraftwerk2 converges in terms of cells overlap and spreading procedure. A good metric to be used as the termination condition is to compare the current placement area with the sum of all cells' area. If they are equal, it means that there is no overlap. Setting $A_\cup$ as the cells' placement area and A the sum of their area, the total overlap percentage can be set as:

$$\Omega = 1 - \frac{A_\cup}{A}. \tag{2.2.25}$$

Usually when Ω is under 0.2 (or 20%) the placement procedure can be terminated and continued by a legalizer.

2.2.7 Summarising the Algorithm

After calculating each one of the three forces, Net force, Hold force and Move force, Kraftwerk2 achieves force equilibrium in each iteration by solving the linear equations:

$$(C_x + \mathring{C}_x)\Delta x = -\mathring{C}_x * \Phi_x \tag{2.2.26}$$

and

$$(C_y + \mathring{C}_y)\Delta y = -\mathring{C}_y * \Phi_y. \tag{2.2.27}$$

The new placement positions are calculated by solving the equations with respect to Δx and Δy and then adding the results to the old position values. The following figure describes the overall algorithm flow.

```
function KRAFTWERK2_PLACER
    Quadratic_Placement();
    while Ω > 20% do
        Create_Demand_Supply_System();
        Calculate_Potential();
        Apply_Bound2Bound_model();
        Create_C_x_C̊_x_Φ_x();
        Create_C_y_C̊_y_Φ_y();
        Solve_linear_equations();
        Update_Placement_Positions();
        Quality_Control();
    end while
end function
```

Figure 2.2: The Kraftwerk2 Algorithm.

As shown in the pseudo-algorithm, the initial part is the Quadratic placement. The construction of the needed matrices is completed in a different thesis, as it will be explained in the next chapter. The iterative part of Kraftwerk2 follows, where as a first step, the Demand and Supply system is created, which is used as an input for the Poisson's equation in order to simulate the electrostatic potential. Then, the target points are calculated based on the gradient of the cells' current positions. By filling the matrices and vectors with the calculated values and solving the linear equation, the new positions can be set to each cell.

2.3 Further Analysis and Implementation Notes

2.3.1 Poisson's Equation

The creation of the Electrostatic Potential via the Poisson's Equation is one of the main concepts of Kraftwerk2, as it leads to the calculation of the Move force. However, to algorithmically solve this equation, a numerical solver has to be used

in order to approximate the solution in a reasonable amount of time. As a result, the input system as well as the output solution have to be discrete. The continuous function of the Demand-Supply System has to be replaced with a grid structure producing an also discrete approximation structure for the potential.

An additional consequence of the approximation approach is the calculation of the gradient values. Since there is no continuous function of the core area, the partial derivative of the potential on each point cannot be calculated accurately.

2.3.2 Bound2Bound Net Model

The Bound2Bound net model is basically removing all inner two-pin connections of a net, taking into account only the ones on the boundary pins. It affects the placement quality by changing only the weight values of C matrix. Hence, it can safely be replaced with a simpler approach. In this thesis, the original clique model will be used, reducing the development time. Kraftwerk2 with the new net model will be set as a reference point, giving as more time to focus on the remaining parts of the algorithm (Demand-Supply system, Poisson's equation etc.) and investigate in depth new heuristics and approaches. The implementation of Bound2Bound and the analysis of the influence of net models will be scheduled as future work.

2.3.3 Quality Control

The default approach of the quality control, namely the scale factor of the cells' weights, $\kappa(\mu)$, as presented in chapter 2, is:

$$\kappa(\mu) = 1 + \tanh\left(\ln\left(\frac{\mu_T}{\mu}\right)\right) \tag{2.3.28}$$

The hyperbolic tangent function, tanh , has a target set limited to the range $(-1, 1)$, meaning that $\kappa(\mu)$ is limited to the range $(0, 2)$. Since this is the only factor to scale up or down the cells' movement based on the target movement, this range may not be enough, considering the 1/M factor of the weights w_i, which creates a very small number in case of a very big circuit. A different approach of the factor $\kappa(\mu)$ will be presented in chapter 4.

2.3.4 Convergence

As explained in chapter 2, the Kraftwerk2 algorithm converges, reducing the overall overlap in each iteration. Again, due to the grid structure, the overlap will be approximate. From the overlap's equation,

$$\Omega = 1 - \frac{A_{\cup}}{A_{mod}}, \tag{2.3.29}$$

A_{mod} was defined as the sum of all cells' area, which is a straight forward calculation. However, $A_{\cup}$ was defined as the current cells' spread area. This would be an easy and fast calculation if the Demand was a continuous function, giving the exact perimeter of the current placement. Without this function, the calculation of the exact placement area will be a very complex and expensive in time procedure.

Chapter 3

Contribution and Development

3.1 Introduction

This chapter focuses on the implementation and the development of the Kraftwerk2 algorithm, including this thesis' contribution. Due to the complexity and size of it, the algorithm was split into two diploma theses: Xanthos Vlachos' work and mine. Both parts were developed in C (programming language) and imported into an already existing EDA tool which provided a visual representation of each circuit's placement and many other features such as the wirelength cost calculation.

This diploma thesis focuses of the calculation of the move force part. Based on the initial form of the Kraftwerk2 algorithm, my contribution consists of the following parts:

- A different construction method of the Demand - Supply System.
- The calculation of the Electrostatic Potential using a fast Poisson Solver.
- An approaching method of the calculation of the cells' gradient.
- The creation of a Gradient Normalization method.
- A new Quality Control factor function.
- A differentiated calculation of the global overlap and the introduction to a new termination condition.

- The construction of the framed supply approach

The diploma thesis of Xanthos Vlachos consists of the construction of the rest parts of the algorithm, including it's main body, the creation of the matrices and vectors of the final equation, the solving procedure and many other optimization parts.

The unification of the two theses created a differentiated edition of Kraftwerk2 capable of successfully placing real benchmarks quite efficiently.

3.2 The New Distribution Model Approach

The construction of the Distribution Model, namely the Demand and Supply System, was one of the most critical issues that had to be fixed via an approximate solution. The reason was that this system would be the input to a solver which uses numerical methods. As a result, the continuous function $D(x, y)$ couldn't be used; instead, a set of discrete set of values had to be implemented. This lead to the dicretezation of the system using a grid structure. Thus, since the initial

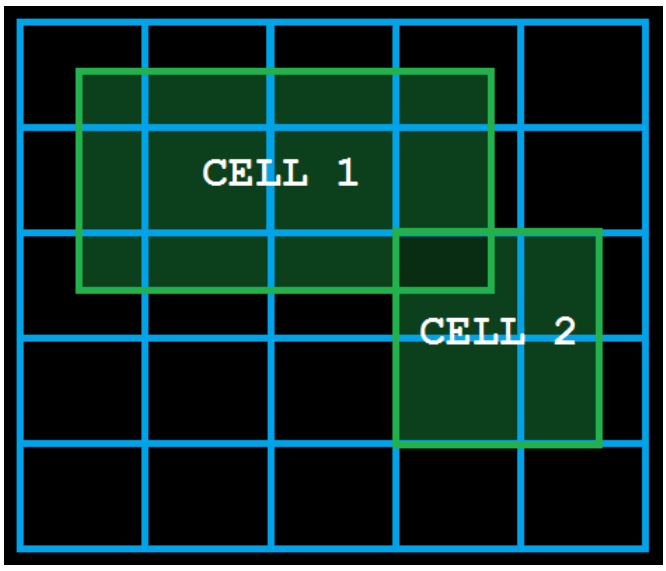

Figure 3.1: Demand-Supply Grid example.

function $D(x, y)$ of the system returns a value given the x and y coordinates of the core area, this area can be partitioned into equally sized pieces, named ***bins***. The

```
function DEMAND_SUPPLY_SYSTEM
    for each Cell k do
        bin_start_i ← cell_start_x[k]/bin_width;
        bin_start_j ← cell_start_y[k]/bin_height;
        bin_end_i ← cell_end_x[k]/bin_width;
        bin_end_j ← cell_start_y[k]/bin_height;
        for Each Bin i from bin_start_i to bin_end_i do
            for Each Bin j from bin_start_j to bin_end_j do
                Demand[i][j] ← Calculated Cell Area to Bin Area Ratio;
            end for
        end for
    end for
    for Each Bin i do
        for Each Bin j do
            Supply[i][j] ← d_sup;
        end for
    end for
    D ← Demand − Supply;
    return D;
end function
```

Figure 3.2: Demand_Supply_System Function.

new approach of the Distribution model can now be stored as a two-dimensional matrix, which returns values given the exact bin. This solution will clearly cause an approximation error, but not a noticeable one in the final result, especially when the number of bins is the right one, something that will be discussed later.

Both the Supply and Demand have to be defined as matrices. In the case of the Demand, each cell will overlap a number of bins. Since we do not refer to points of the core area anymore, these bins cannot be set to one. Hereafter, each bin will be set to the ratio of the overlapping area of the cell to the area of the bin. The location of the start of the cell (top left corner) and the location of the end of the

cell (bottom right corner) are translated to bin_start and bin_end and the demand for each bin of that range is calculated. For example, in Figure 3.1 the CELL 1 overlaps 12 bins, where in particular, the top left bin is set to 0.25, the one on it's right to 0.5, etc.. The bins that are overlapped by 100% from a cell are set to 1 and and the ones that are overlapped from more than one cell are set to the sum of their values.

In the case of the Supply, an easier approach is implemented, setting every bin to a constant value d_{sup} as shown in the Chapter 2. The overall construction of the system is enclosed to the *Demand_Supply_System* function as presented in Figure 3.2. A visual representation of D is shown in Figure 3.3,which corresponds to the initial quadratic placement of a circuit. Each colored box represents every bin's value.

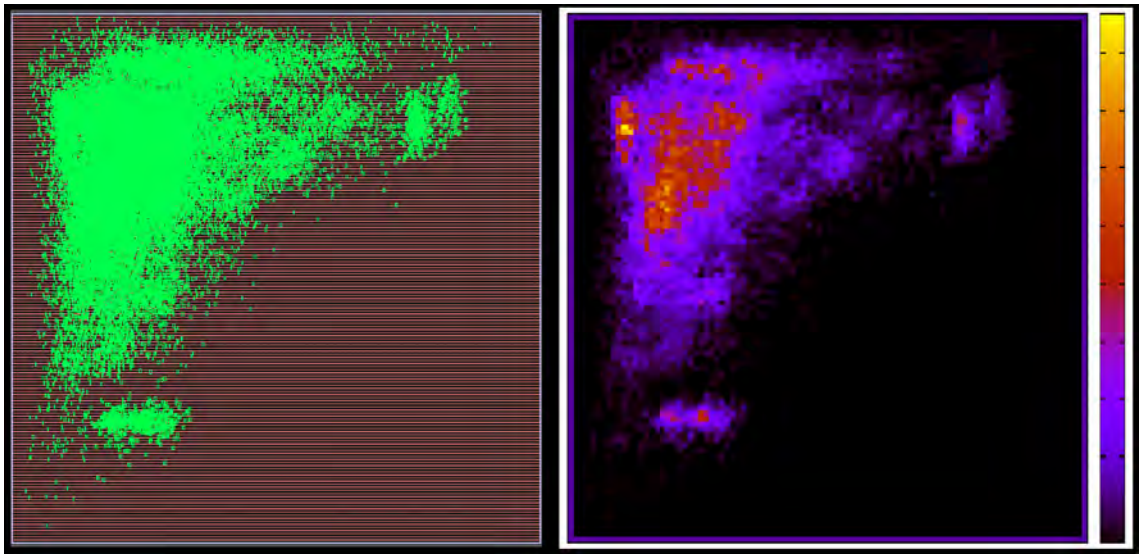

Figure 3.3: Left: Initial Quadratic Placement. Right: Demand-Supply System.

3.3 Electrostatic Potential Φ

Since D is created, the computation of the electrostatic potential is a more automated procedure. For this procedure, the Fast Poisson Solver Routine from Intel's

Math Kernel Library (MKL) was used, offering the most suitable and fast solver for this occasion.

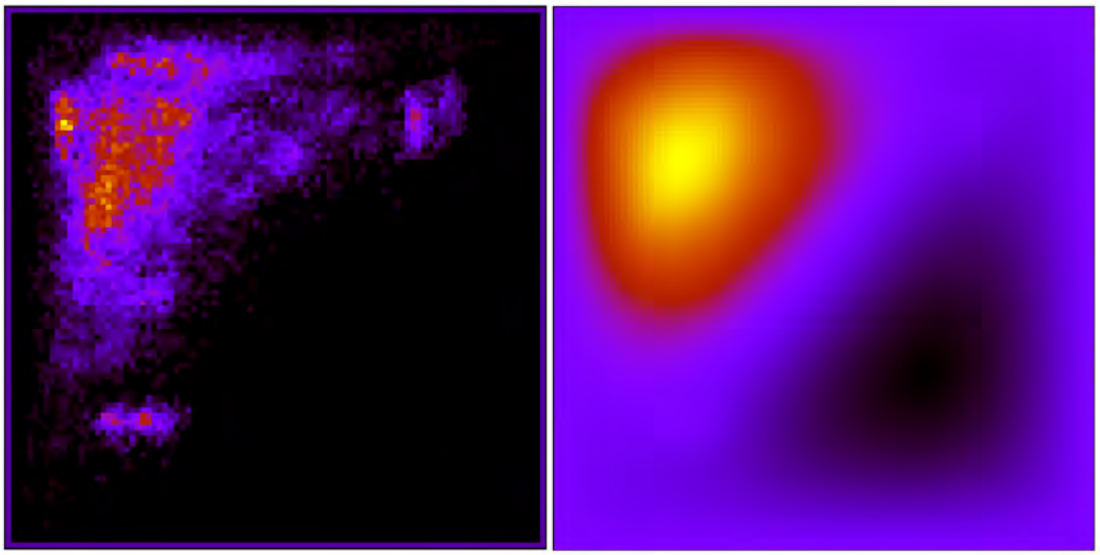

Figure 3.4: Left: Demand-Supply System. Right: Potential Φ.

The most difficult part of this section was to choose the boundary conditions, which define the value of each position of the boundary surfaces; in our case, the perimeter of the chip. The ***Dirichlet*** boundary condition sets the exact value D at all of the boundary surfaces. The ***Neumann*** boundary condition sets the ∇D at the the boundary surfaces.

Each case was tested and the experiment showed that ***Neumann*** condition results a better solution, as presented in the next chapter. Figure 3.4 shows the visual representation of the Demand - Supply System with the corresponding Potential Φ of a placement example.

3.4 Gradient

Given the electrostatic potential matrix Φ, the final step for the calculation of the target points is the creation of the gradients vectors. As shown in chapter 2, each element of the vectors equals to the partial derivative of the potential at the exact position of each cell, for the x and y dimensions. Due to the grid structure ap-

proach that is being used for the potential, these gradients have to be calculated approximately.

From it's definition, the partial derivative can be calculated as shown bellow:

$$\begin{aligned} \Phi_x &= \frac{\partial \Phi}{\partial x} = \lim_{h \to 0} \frac{\Phi(x+h, y) - \Phi(x, y)}{h} \\ \Phi_y &= \frac{\partial \Phi}{\partial y} = \lim_{h \to 0} \frac{\Phi(x, y+h) - \Phi(x, y)}{h} \end{aligned} \tag{3.4.1}$$

or

$$\begin{aligned} \Phi_x &= \frac{\partial \Phi}{\partial x} = \lim_{h \to 0} \frac{\Phi(x, y) - \Phi(x-h, y)}{h} \\ \Phi_y &= \frac{\partial \Phi}{\partial y} = \lim_{h \to 0} \frac{\Phi(x, y) - \Phi(x, y-h)}{h} \end{aligned} \tag{3.4.2}$$

Since the smallest value of h can only be the width and the height of the bin, the equations of the gradients can be transformed as follows:

$$\begin{aligned} \Phi_x &= \frac{\Phi[i+1, j] - \Phi[i, j]}{hx} \\ \Phi_y &= \frac{\Phi[i, j+1] - \Phi[i, j]}{hy} \end{aligned} \tag{3.4.3}$$

or

$$\begin{aligned} \Phi_x &= \frac{\Phi[i, j] - \Phi[i-1, j]}{hx} \\ \Phi_y &= \frac{\Phi[i, j] - \Phi[i, j-1]}{hy} \end{aligned} \tag{3.4.4}$$

where i, j are each bin's matrix location values and hx, hy are the bins' width and height, respectively. Calculating the average value of the two approaches, the gradient vectors can be created with a small approximation error.

3.4.1 Gradient Normalization

The values of the gradients play a key role to the final solution. As explained in Chapter 2, the x and y position of the target point for each cell is calculated via the expression

$$\begin{aligned} \mathring{x}_i &= x'_i - \frac{\partial}{\partial x} \Phi(x, y)|_{(x'_i, y'_i)}, \\ \mathring{y}_i &= y'_i - \frac{\partial}{\partial y} \Phi(x, y)|_{(x'_i, y'_i)}. \end{aligned} \tag{3.4.5}$$

Scaling up or down the calculated gradients will affect the target points which will result a slower of faster spreading procedure. By calculating the average value of the gradients of the current iteration and given the desirable value from the user, a scale factor can be extracted and multiplied to every gradient:

$$\begin{aligned}\frac{\partial}{\partial x}\Phi(x,y)|_{(x'_i,y'_i)} &= \mathbf{s_x} * \frac{\partial}{\partial x}\Phi(x,y)|_{(x'_i,y'_i)},\\ \frac{\partial}{\partial y}\Phi(x,y)|_{(x'_i,y'_i)} &= \mathbf{s_y} * \frac{\partial}{\partial y}\Phi(x,y)|_{(x'_i,y'_i)}.\end{aligned} \tag{3.4.6}$$

In addition, this factor can also be multiplied by the current overlap, since the spreading procedure must be more detailed as the algorithm converges. Specifically, the scale factor is calculated as show bellow:

$$\begin{aligned}\mathbf{s_x} &= \frac{\mathbf{p} * \mathbf{Core\_Size\_X} * \mathbf{Overlap}}{\mathbf{Average\_Gradient\_X}},\\ \mathbf{s_y} &= \frac{\mathbf{p} * \mathbf{Core\_Size\_Y} * \mathbf{Overlap}}{\mathbf{Average\_Gradient\_Y}},\end{aligned} \tag{3.4.7}$$

where p is the desired percentage of the core dimension which the user wants the average gradient to be equal to. For the purpose of the experiments, p will be set to 0,1.

3.5 Quality Control

As explained in chapter 3, the initial edition of the $\kappa(\mu)$ factor for the quality control is limited to the range (0, 2). For that reason, an other approach is also implemented to set the upper limit to infinity using the ***Reciprocal Function***:

$$\kappa(\mu) = \frac{1}{\frac{\mu}{\mu_T}} \tag{3.5.8}$$

Besides the upper limit, the two approaches have a quite similar set of values, as shown in Figure 3.5.

The second approach will prove important, as it scales up the small step size of the first one. Their comparison is shown in the Experiments section.

The value of the target movement μ_T is set by the user, however a default value has to be chosen. In the experimental procedure this variable will be tested.

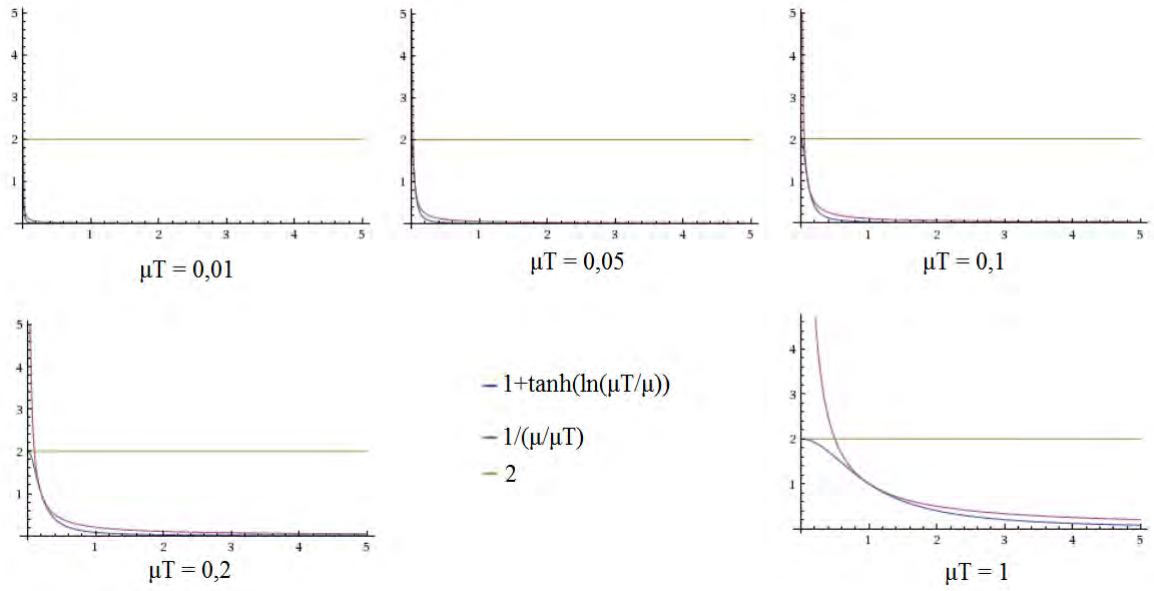

Figure 3.5: Graphical representation of the two quality control approaches.

3.6 Termination Condition

The already created grid structure can be used to calculate the overlap with a small approximation error. Namely, every bin's value is checked and if it is greater that 1, the whole bin's area is added to $A_{\cup}$. Instead, when the bin's value is lower than 1, it's area is multiplied with that value.

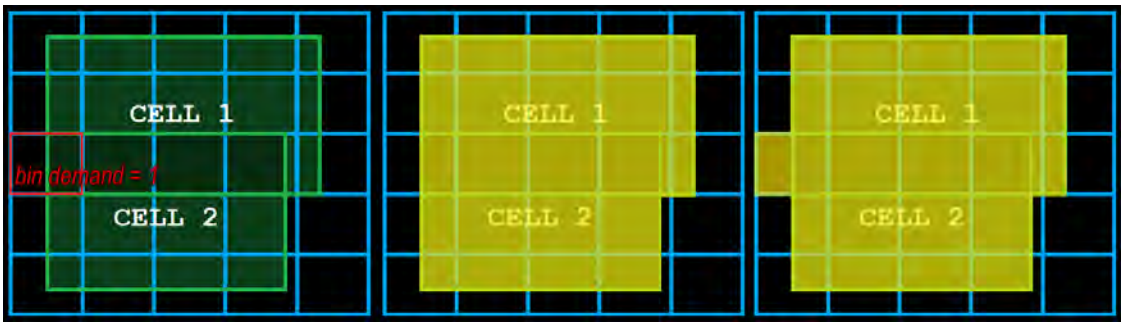

Figure 3.6: Cause of the overlap approximation error. Left: The bin in the red rectangle has a demand value of 1. Middle: The actual spread area of the cells. Right: The calculated spread area of the cells.

That way, when a cell overlaps, for example, only half of the area of a bin then

the half of the bin's area is added to $A_{\cup}$. The problem with this approach occurs when multiple cells overlap with each other in one bin's area. When all of them are placed in one small region of it, the bin's demand value will be greater than the ration of the area that has been overlapped. Figure 3.6 shows a simple example of this case, where the two cells overlap only half of the area of the leftmost and third from top bin, and yet, the bin's demand value is 1. This approximation error is unavoidable but decreases when the size of the bins is small enough. In any case, the error will be greater that zero, causing a faster termination time and spreading the cells slightly less than the set value.

Taking advantage of the bins structure, another termination condition can be the average bin density:

$$Density = \frac{1}{NX * NY} * \sum_{i,j=1}^{i=NX,j=NY} |Demand[i][j] - 1|, \tag{3.6.9}$$

where NX, NY are the size of the structure for the x and y coordinates. The value of this metric reduces as the cells spread over the core area and the number of empty or much overlapped bins is less.

3.7 Overall Algorithm Flow

The main structure of the algorithm does not change much. The differences are located in each function, where the Bound2Bound net model is now replaced, the Demand-supply system, the electrostatic potential and the gradients are now discrete, the gradient normalization feature is added and the Quality control function is changed. Finally, the termination condition now includes the Total density, where its termination value is heuristically set to 0.7.

function NEW_KRAFTWERK2_PLACER

Quadratic_Placement();

while $(\Omega > 20)$ *Or* $(Density > 0.7)\%$ **do**

Demand_Supply_System();

Calculate_Potential();

Calculate_Gradients();

Gradients_Normalization();

*Create_*C_x*_*$\mathring{C}_x$*_*Φ_x();

*Create_*C_y*_*$\mathring{C}_y$*_*Φ_y();

Solve_linear_equations();

Update_Placement_Positions();

Quality_Control();

end while

end function

Figure 3.7: The New Kraftwerk2 Algorithm.

Chapter 4

Experiments

4.1 Experimental Methodology

For the experimental procedure, eight industrial and academic benchmarks were used covering a range of movable components (cells) number from 716 to 219263. The main comparison metrics were the global overlap and density, the total wire-length cost, the number of iterations and the total execution time. Equally important was the observation of the visual representation of each benchmark, since it is impossible to keep track of the movement of each cell in such big examples. Table 4.1 shows the basic characteristics of these Benchmarks.

Benchmark	# Cells	Core Width	Core Height
BM1	716	29.88	29.952
BM2	2343	48.06	47.808
BM3	18793	108.72	108.864
BM4 - bridge32_1	30675	319	320
BM5 - fft	32281	342	342
BM6 - cordic_I4	41601	360.6	360
BM7 - matrix_mult	155325	696.8	696
BM8 - b19_fast	219263	1233.8	1234

Table 4.1: Characteristics of the Benchmarks used for the experiments

For the comparison purposes of each feature that will be tested, a default setup will be set. Hence, these features will be considered as selected, unless otherwise specified. Namely, this setup consists of the following:

- Neumann Boundary Condition for the Electrostatic Potential
- 20% Overlap as Termination Condition
- $\kappa(\mu) = \frac{1}{\frac{\mu}{\mu_T}}$ as Quality Control scale factor
- No Gradient Normalization
- $\frac{corewidth}{sitewidth}$ and $\frac{coreheight}{sitewidth}$ as number of bins horizontally and vertically in the grid structure
- 5% of the core width as target movement (μ_T)
- 50% utilization
- Aspect ratio 1

	Boundary Condition	QC	Grad. Normalization	Termination Condition
Setup 0	Neumann (Case 1)	Reciprocal	No	20% Overl.
Setup 1.1	Dirichlet (Case 1)	Reciprocal	No	20% Overl.
Setup 1.2	Dirichlet (Case 2)	Reciprocal	No	20% Overl.
Setup 1.3	Neumann (Case 2)	Reciprocal	No	20% Overl.
Setup 2	Neumann (Case 1)	Hyperbolic Tangent	No	20% Overl.
Setup 3	Neumann (Case 1)	Reciprocal	Yes	20% Overl.
Setup 4	Neumann (Case 1)	Reciprocal	No	70% Dens.

Table 4.2: Characteristics of the Setups used for the experiments

By changing one feature at a time, a number of setups is created and defined above. Setup 0 is the reference point, as explained previously, in which the features

have been chosen in such a way that they lead to a more effective, stable and, at the same time, close to the original Kraftwerk2 approach. Hence, no Gradient Normalization is applied, the termination condition is the default one with the same overlap percentage, but the new scale factor of the Quality Control is used because it reduces the execution time drastically. Furthermore, the global utilization is set to 50%, since the quality of the spreading procedure is important. Hence, overlap must be reduced but by kipping cells together. A big utilization percentage will show if cells spread all over the core area. In addition, the bins' width and height is set to the site width, which the smallest fraction on a chip, and the target movement is heuristically set to 5% of the core width. Finally, the Neumann boundary conditions are selected, as they give a more qualitative result and the Aspect Ration is set to 1 since it is excluded from the experimental procedure.

In Table 4.2 the constrains of each setup are defined.

4.2 Results

4.2.1 Initial Approach - Setup 0

The execution time mostly depends on the number of iterations. The initial QP is the factor that affects that number the most. Besides that, the number of connections between cells and pins may cause an increase in the value of net forces, slowing down the spreading procedure. For example, BM4 is 1.6 times bigger than BM3, with about the same QP overlap but with 8.6 times less QP wirelength cost. As a result, BM4 needs less placement iterations.

Benchmark	QP WL Cost	QP Overlap	QP Density	Bins (width*height)
BM1	24706.936	0.7352	1.2344	332*333
BM2	104045.845	0.7515	1.2528	534*531
BM3	453795.431	0.8928	1.3925	1208*1210
BM4	52827.599	0.9289	1.4277	1595*1600
BM5	5510957.27	0.5935	1.0935	1710*1710
BM6	201502.42	0.9722	1.4728	1803*1800
BM7	21309754.06	0.7531	1.2533	3484*3480
BM8	1514712.412	0.9797	1.4794	12338*12340

Table 4.3: Results before the spreading procedure (QP).

Table 4.3 and 4.4 show the results of Setup 0 as defined previously before and after the spreading procedure. These results will be useful as a reference point for the rest test cases.

Benchmark	Overlap	Density	WL Cost	Iterations	Time
BM1	0.1967	0.6916	36359.411	53	17 sec
BM2	0.1997	0.6981	155517.039	112	56 sec
BM3	0.1994	0.6997	1255921.204	289	7 min 14 sec
BM4	0.1999	0.7007	1591264.039	206	4 min 32 sec
BM5	0.1999	0.7000	8222871.877	227	3 min 57 sec
BM6	0.1996	0.6992	1283282.740	205	5 min 10 sec
BM7	0.1999	0.6996	32498874.957	439	1 h 5 min 38 sec
BM8	0.1998	0.6999	27364908.245	602	5 h 16 min 55 sec

Table 4.4: Final results of Setup 0

In the following Figures 4.1, 4.2 and 4.3, the overall visual illustration of the placement, as well as the Average movement, Target movement, Overlap and Density are shown in detail for 4 representable Benchmarks (BM1, BM3, BM6 and BM7).

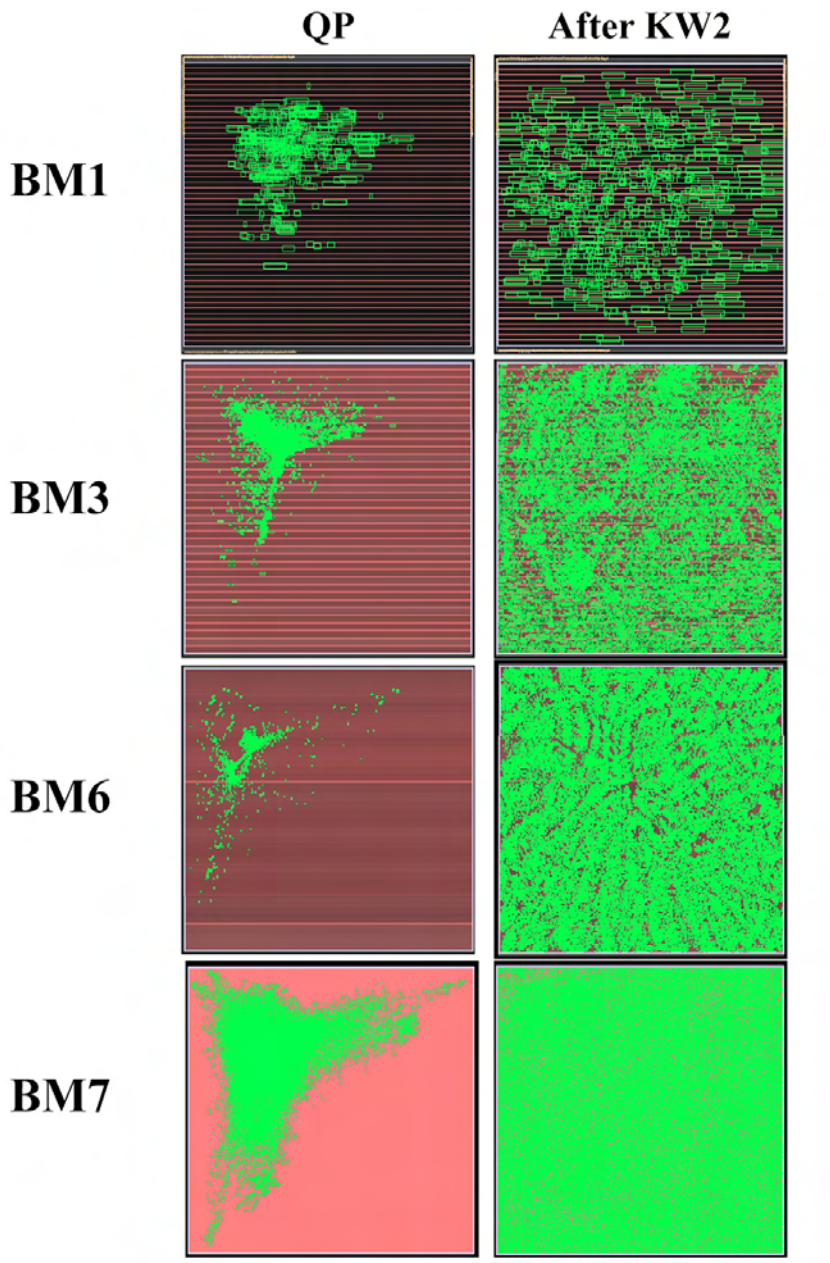

Figure 4.1: Visual representation of the initial and the final placement result.

In Figure 4.2 the Average movement is reduced during the iterative procedure. It can be observed that in BM1 the movement is reduced smoothly in contrast with the rest of the benchmarks, where the first 20 iterations result much more movement. This is due to the initial quadratic placement. In BM1 the QP overlap is relatively small and the maximum density is quite less than the density on the QP of the rest of the benchmarks. This results a smoother potential and, hence, smaller gradient

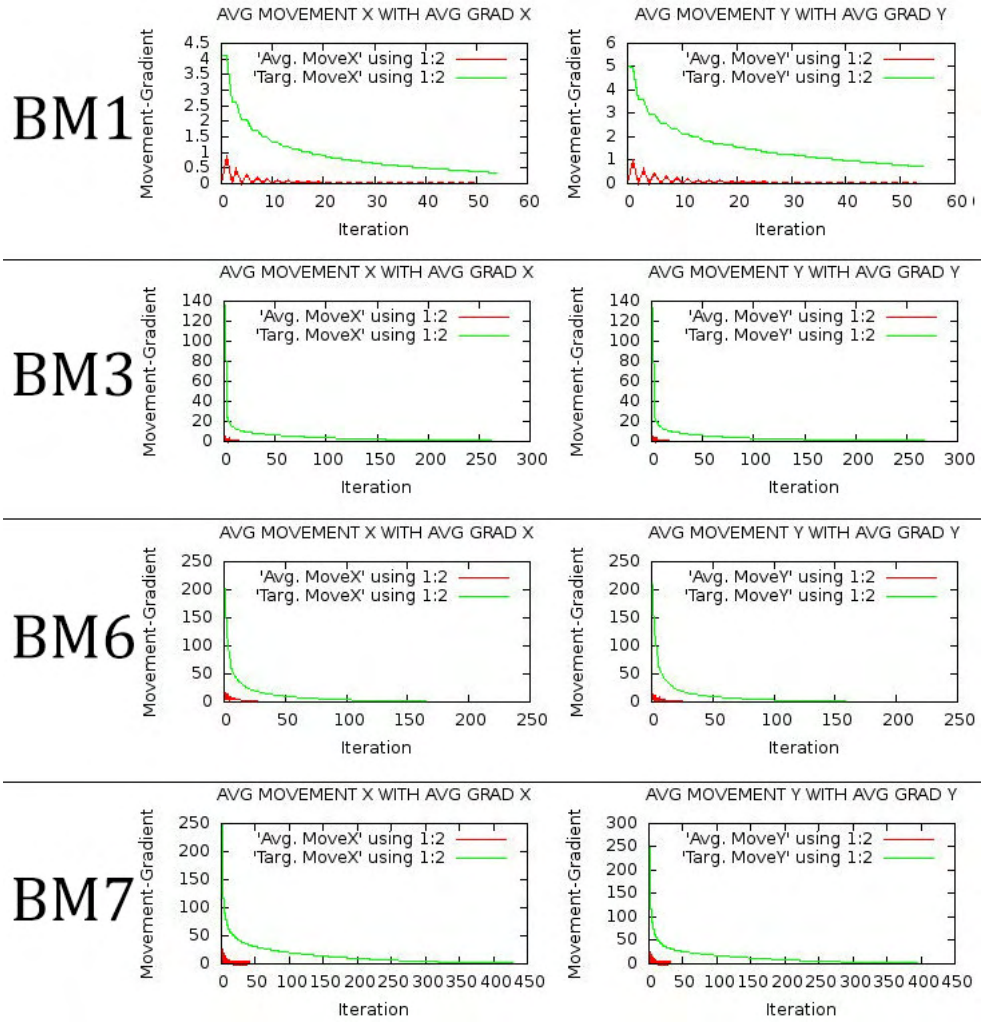

Figure 4.2: Average cells movement and average target movement in each iteration graph.

values.

Figure 4.3 shows clearly that the overlap and density values change in the same way during the iterations.

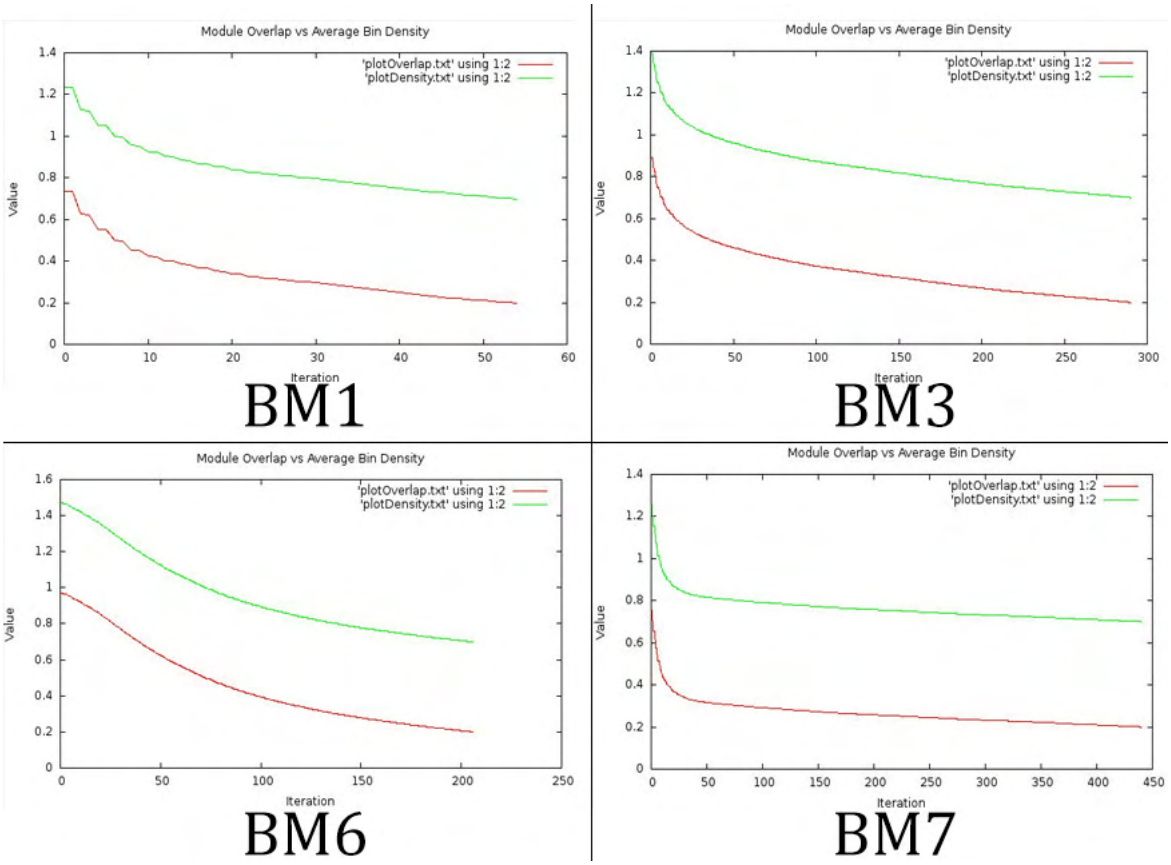

Figure 4.3: Reduction of the overlap and density over the iterations.

4.2.2 Boundary Condition - Setup 1

To choose the Boundary Condition, both Dirichlet and Neumann were tested, setting the following values in four different cases:

- Dirichlet Boundary Condition
  - $u(x, y) = 0$
  - $u(x, y) = \Phi(x, y)$
- Neumann Boundary Condition
  - $\frac{\partial u}{\partial x}(x, y) = 0,$
    $\frac{\partial u}{\partial y}(x, y) = 0$
  - $\frac{\partial u}{\partial x}(x, y) = \Phi(x, y),$
    $\frac{\partial u}{\partial y}(x, y) = \Phi(x, y)$

In the initial approach, the first case of the Neumann Boundary Condition has been selected.

	Dirichlet Case 1			Dirichlet Case 2			Neumann Case 1			Neumann Case 2		
	Conv	WL	Iter	Conv	WL	Iter	Conv	WL	Iter	Conv	WL	Iter
BM1	No	-	-	No	-	-	Yes	36359.411	53	Yes	36084.328	49
BM2	No	-	-	No	-	-	Yes	155400.919	111	Yes	156556.298	110
BM3	No	-	-	No	-	-	Yes	1255921.204	289	Yes	1261405.079	306
BM4	No	-	-	No	-	-	Yes	1591264.039	206	Yes	1592354.556	208
BM5	No	-	-	No	-	-	Yes	8222871.877	227	Yes	8191609.302	219
BM6	No	-	-	No	-	-	Yes	1283282.740	205	Yes	1283619.262	205
BM7	No	-	-	No	-	-	Yes	32498874.957	439	Yes	32472792.421	445
BM8	No	-	-	No	-	-	Yes	27364908.245	602	Yes	27364908.245	602

Table 4.5: Final results for each case of the Boundary Conditions

As shown in Table 4.5, the cases of the Dirichlet Boundary Condition didn't converge while the cases of Neumann converged almost similarly.

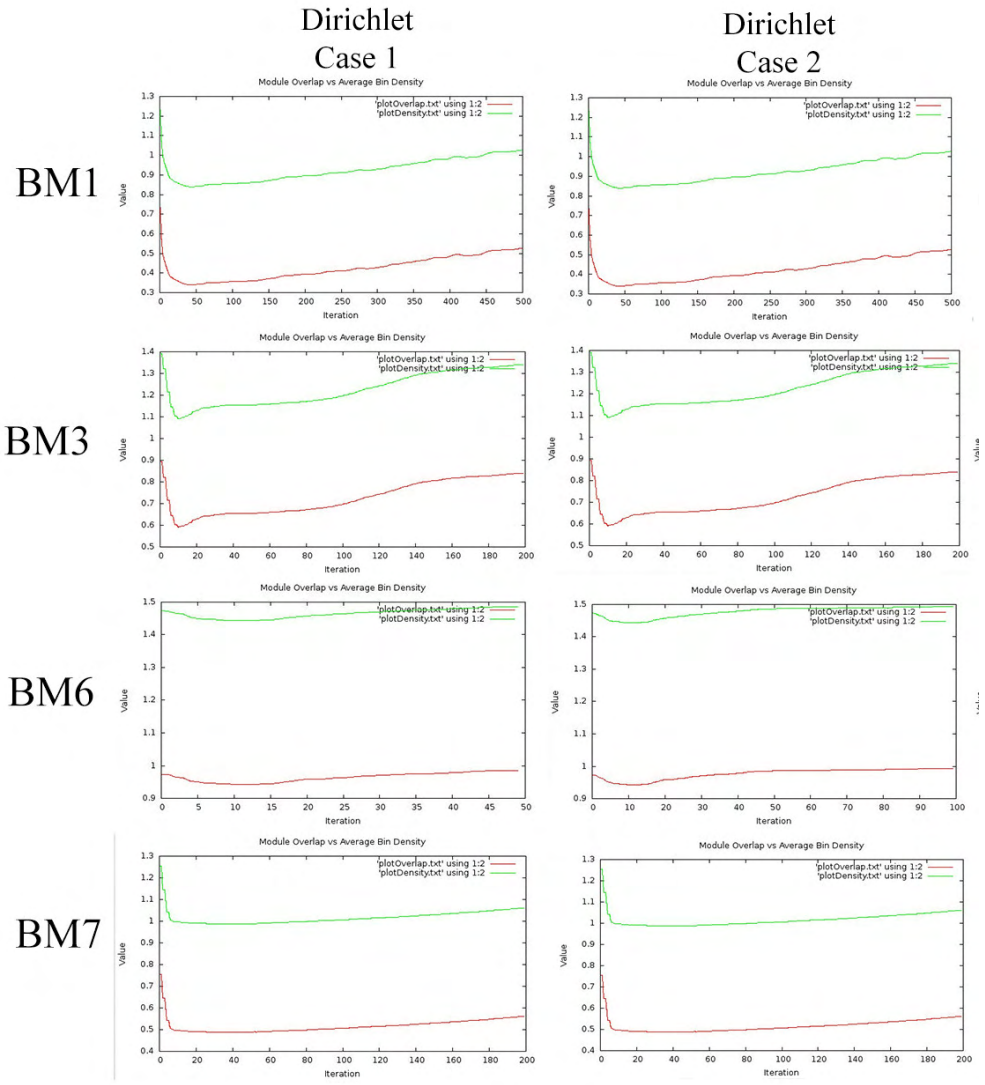

Figure 4.4: Progress of the overlap and density over the iterations.

The graphs in Figure 4.4 confirm this lack of convergence, since the overlap and density start to increase again at some point. Figure 4.5 shows the small difference between the two Neumann cases, where the first case results a wider spreading result.

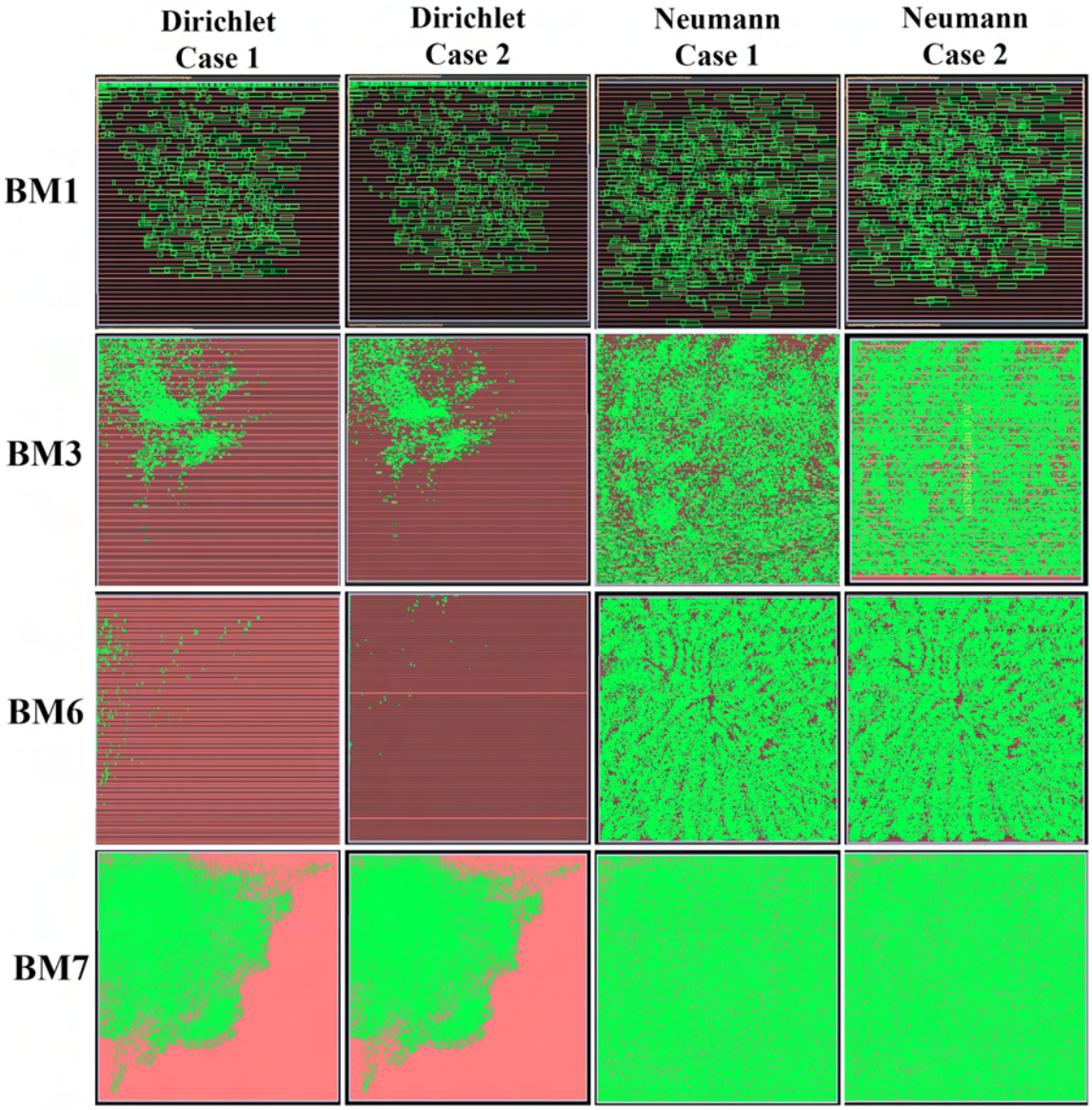

Figure 4.5: Placements for each boundary condition case.

4.2.3 Quality Control - Setup 2

To test the new Quality Control factor approach, $\kappa(\mu)$, Table 4.6 presents the comparison with the default Kraftwerk2's approach. It is important to note that tests have an upper limit of 10000 iterations. With the default approach, almost all the tests reached that limit, so their reached overlap will be the common terminate point.

	Setup 2			Setup 0		
	Overlap	WL	Iteration	Overlap	WL	Iteration
BM1	0.1999	34960.535	1320	0.1967	36359.411	53
BM2	0.1999	149372.275	4301	0.1997	155517.039	112
BM3	0.2674	1145178.781	10000	0.2668	118756.447	205
BM4	0.2111	1574902.382	10000	0.2103	1577093.060	197
BM5	0.2627	746665.050	10000	0.2625	754584.545	55
BM6	0.2331	1251864.485	10000	0.2330	1250975.298	179

Table 4.6: Final results for each case of the Boundary Conditions

From the results it can be concluded that the new Quality Control factor causes a significant reduction of the number of the iterations, sacrificing a very small percentage of the total wirelength.

Tables 4.7 and 4.8 present the influence of the μ_T's value in the benchmarks BM1, BM3, BM6 and BM7. The value of μ_T if set to the percentage of the core dimension for x and y coordinates. In these experiments, different values of this percentage were tested.

μ_T (% of Core Dimensions)	BM1	BM3	BM6	BM7
0.05	36359.411	1255921.204	1283282.740	32498874.957
0.1	36395.873	1287118.937	1282041.774	32804508.225
0.2	37251.704	1343340.874	1285275.455	33499937.929
0.5	39113.120	1465108.506	1333792.799	35444375.967
1	40440.871	1712020.870	1467538.682	37931398.839
5	43114.379	2329551.785	1583137.325	47240035.201
10	43658.307	2670956.842	1682853.637	51923138.276

Table 4.7: Influence of the μ_T (given as percentage of the Core Dimensions X and Y) to the total wirelength cost.

μ_T (% of Core Dimensions)	BM1	BM3	BM6	BM7
0.05	56	289	205	439
0.1	29	196	145	290
0.2	19	126	104	191
0.5	13	71	68	108
1	10	46	50	73
5	8	21	24	47
10	7	17	18	38

Table 4.8: Influence of the μ_T (given as percentage of the Core Dimensions X and Y) to the number of placement iterations.

Table 4.9 shows the major reduction of the number of placement iterations with the increase of the total wirelegth cost by a much smaller percentage.

	BM1		BM3	
	WL	Iter	WL	Iter
0.05 to 0.1	0.1%	-48.2%	2.48%	-32.17%
0.1 to 0.2	2.35%	-34.48%	4.36%	-35.71%
0.2 to 0.5	4.99%	-31.57%	9.06%	-43.65%
0.5 to 1	3.39%	-23.07%	16.85%	-35.21%
1 to 5	6.61%	-20%	36.07%	-54.34%
5 to 10	1.26%	-12.5%	14.65%	-19.04%
	BM6		BM7	
	WL	Iter	WL	Iter
0.05 to 0.1	-0.09%	-29.26%	0.94%	-33.94%
0.1 to 0.2	0.25%	-28.27%	2.12%	-34.13%
0.2 to 0.5	3.77%	-34.61%	5.8%	-43.45%
0.5 to 1	10.02%	-26.47%	7.01%	-32.40%
1 to 5	7.87%	-52%	24.54%	-35.61%
5 to 10	6.29%	-25%	9.91%	-19.14%

Table 4.9: Percentage of increase/decrease of the wirelength and number of iterations from the increase of the μ_T (% of the Core Dimensions X and Y).

4.2.4 Gradient Normalization - Setup 3

The effect of the new Gradient Normalization is more obvious in large scale benchmarks. Figure 4.6 bellow shows the difference between the initial setup and the setup 3 with the new feature. Note that both placements of each benchmark are terminated at the same iteration.

4.2.5 Termination Condition - Setup 4

As the table 4.4 shows, the density values of benchmark placements are very close to 0.7 when the overlap value is 0.2. This means that the termination condition can be safely replaced . The critical difference of the density condition is that it has no estimation error, as explained in chapter 4. Therefore, it would be a safer option to use in bigger or more complicated benchmarks in order to avoid excessive numbers of iterations.

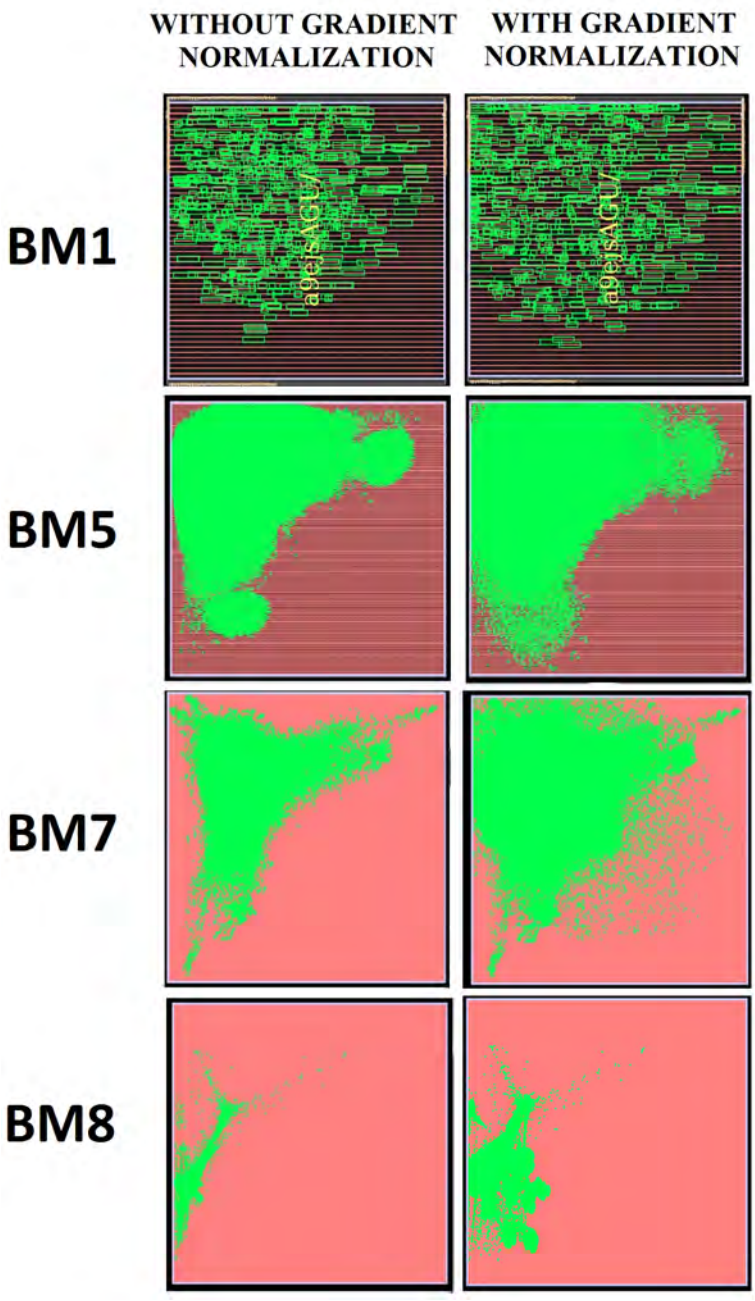

Figure 4.6: Gradient normalization results.

Chapter 5

Conclusions and Future Work

Within this thesis, a successful analysis, implementation and improvement of Kraftwerk2 has been completed. It became clear that it has a lot of potentials to be improved. In combination with the new enhancements, the algorithm is capable of facing most circuits and offer a very fast but qualitative result by adjusting each user-given constrain.

The algorithm can be extended in many levels in order to become a more complete placer. Specifically, cluster support can be included, as well as 3D and timing-driven placement. Another concept is to replace Poisson's equation with a blurring algorithm in order to achieve faster results. To complete the placer, a net model must be implemented and compared with the Bound2Bound. Execution time can be reduced by placing parts on a chip simultaneously in floorplanned designs.

Finally, Krafterk2 must be combined with detailed-placer/legalizer, since it allows a percentage of overlap in the result.

Bibliography

[1] Peter Spindler, *Efficient Quadratic Placement of VLSI Circuits*, Technical University of Munchen, 2007.

[2] H. Eisenmann and F. M. Johannes, *Generic global placement and floorplanning*, Proc. ACM/IEEE DAC, Jun. 1998, pp. 269–274.

[3] Peter Spindler, Ulf Schlichtmann, Member, IEEE, and Frank M. Johannes, *Kraftwerk2—A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model*, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 27, NO. 8, AUGUST 2008.

[4] Sung Kyu Lim, *Practical Problems in VLSI Physical Design Automation* Springer Science+Business Media, Jul 31, 2008.

[5] Charles J. Alpert, Dinesh P. Mehta, Sachin S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*, Auerbach Publications, November 12, 2008.

[6] Kahng, A.B., Lienig, J., Markov, I.L., Hu, J, *VLSI Physical Design: From Graph Partitioning to Timing Closure* Springer Science+Business Media, 2011.