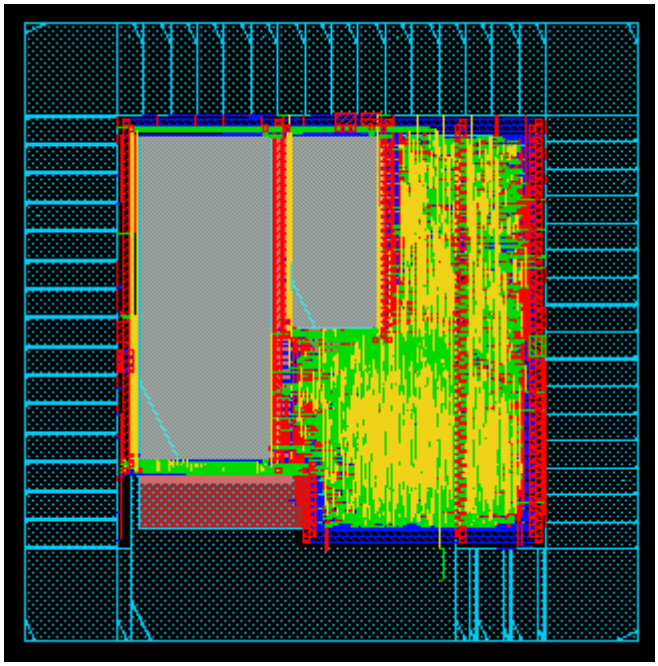
UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF COMPUTER AND COMMUNICATIONS ENGINEERING

Wirelength and Power-Aware driven standard-cell placement

Master of Science thesis



Arvanitakis Ioannis

18/02/2013

Acknowledgements

Starting writing this thesis I would like to thank all everyone who made it possible. First of all, I would like to thank Professor George Stamoulis for his admittance and for being a great supervisor, as the patience, the persistence, the advice and the observations of him were very valuable throughout the duration of this thesis.

Furthermore, I would like to thank Assistant Professors Nestoras Eumorfopoulos and Panagiota Tsompanopoulou for their support and advice on their area of interest at this thesis.

I am also grateful to Associate Professor Ioannis Moondanos and Christos Sotiriou for being helpful for this thesis and for teaching me the full flow of industrial tools, thus frond-end and back-end flow.

Moreover I would like to thank my colleague Irene Psirra. The wirelength driven placement described in this thesis is a product of our cooperation. Also I would like to thank her, for being helpful and cooperative throughout the whole studies, as we have cooperated in many projects, as well as for her support and encouragement.

Many thanks to everybody at VEDAlab, as the exchange of ideas and their support were really helpful for my thesis. Namely I would like to thank Dr. Antonios Dadaliaris and PhD candidate Konstantinos Daloukas.

Many thanks to Sung Kyu Lim for giving us permission to use some examples of placement algorithms, from his book “Practical problems in VLSI physical design automation”.

Also I would like to thank my friends, for their love and support, because if there weren’t them everything would be harder and less pleasant.

Last, but not least, I would like to thank my family for being supportive all these years of my studies. I would like them to know that I am really grateful for everything they have done for me.

Contents

1. Introduction

The research on the field of digital circuits through the decades, from the construction of the first computer to nowadays, has revised basic target many times. Primarily, the target of the scientist was the functionality of the circuits but later changed to the area, time, power etc of the circuit.

In the past meeting area, timing, or power constraints was achievable but now with the technological advancement, competition and the growing complexity of the chips, make the design process a compound, interlaced and time-consuming procedure. In this process, many optimization goals and tight constraints must be addressed.

A key task in chip design is placement, in which the positions of the modules have to be determined in a given chip area. Placement has a direct impact of the total wirelength needed to interconnect the modules, as well as and on the performance of the chip for timing and power.

For these reasons placement's requirements have been subjected to many modifications, as placement tools became an essential component in design flow at many stages.

Because of all these reasons, powerful, flexible and fast placement algorithms are of particular interest more than ever.

In this thesis the history of placement techniques and the state-of-art techniques are referenced and new placement algorithm, as well as, thoughts about power-aware driven placement and other considerations about placement are introduced.

2. Preliminaries

In this chapter it is introduced the basic terminology of this thesis. Before the problem's description we will present the basic flow of the chip design.

2.1. Definitions

Integrated Circuit (IC)**:** or monolithic integrated circuit (also referred to as chip, or microchip) is an electronic circuit manufactured by the patterned diffusion of trace elements into the surface of a thin substrate of semiconductor material. [wikipedia]

Very-Large-Scale Integration (VLSI): is the process of creating integrated circuits by combining thousands of transistors into a single chip. [wikipedia]

Application-Specific Integrated Circuit (ASIC): is an IC customized for a particular use, rather than intended for general-purpose use. [wikipedia]

Standard-cell library: is a collection of low-level logic functions such as AND, OR, INVERT, flip-flops, latches, and buffers. These cells are realized as fixed-height, variable-width full-custom cells. The cells are fixed-height, variable-width full-custom. The fixed-height enables them to be placed in rows. [wikipedia]

Integrated circuit layout: also known IC layout, IC mask layout, or mask design, is the representation of an integrated circuit in terms of planar geometric shapes which correspond to the patterns of metal, oxide, or semiconductor layers that make up the components of the integrated circuit. [wikipedia]

Core area/Die area: Core area is the area of silicon needed for the cell placement. Die area is the whole silicon area of the chip, as these areas may differ.

Module: is defined as a block of cells, other modules or macros. The most used style is the standard cell layout because of its fixed height of the modules that decrease the time complexity and memory allocated for placement algorithms, which for big circuits of 10000000 gates is critical.
There are five major styles of layout:

1. *Gate array:* The gate array design consists of prefabricated silicon with identical modules distributed evenly on the real-estate. The function of a module is determined solely by its connections. Therefore the entire logic is determined by the wires. Space has been reserved for routing
2. *Sea-of-gates:* The sea-of-gates layout is similar to gate array, but no space is reserved for routing. Instead the entire real estate has been filled with preferable transistors. Some of the transistors become unusable however since space must still be allocated for routing.
3. *Standard-cell:* is a collection of low-level logic functions such as AND, OR, INVERT, flip-flops, latches, and buffers. These cells are realized as fixed-height, variable-width full-custom cells. The cells are fixed-height, variable-

width full-custom. The fixed-height enables them to be placed in rows. Originally routing was done between rows but multilayer technology now allows for routing anywhere on the real-estate.

4. *Mixed-cell:* The mixed-cell model is similar to standard-cell layout, but allows large modules in the layout which may very is height and width
5. *General-cell(Macros):*The final layout style which is also the only full-custom is the general-cell layout style. In this case modules are allowed any size and position on the real estate. [3]

Overlap: Two modules overlap with respect to placement if the upper right corner coordinates of the first are smaller than the lower left corner of the second.

Legal placement: A placement is legal when the following constraints are met:

- There is no overlap between modules
- All modules are within the core area

2.2. Computer Aided Design

Computer Aided Design (CAD) is the use of computer systems to design detailed physical objects, through the entire research and development process, thus for the creation, modification, analysis, optimization and final draw of a design. CAD software was created to assist the designer, deal with more complex designs, reduce their faults, and decrease the completion time. Moreover, the designer is allowed to keep documentations and create databases for manufacturing. CAD output is often in the form of electronic files for print or machine operations.

CAD involves all the information needed for the manufacturing process, such as shapes, materials, processes, dimensions and tolerances according to application-specific conventions. Furthermore it may be used to design curves and figures in two-dimensional (2D) or three-dimensional (3D) space.

Picture 1: 2D and 3D CAD model (wikipedia)

Nowadays, the number of industries turning to CAD is growing, because of its benefits such as lower cost of product development and a shortened design cycle. CAD software is extensively used in many applications such as automotive, shipbuilding, aerospace and microelectronic industries, industrial and architectural design.

Those are the reasons of why the computer aided design has become an especially important technology within the scope of computer-aided technologies. It is one of many tools used by engineers and designers and is used in many ways depending on the profession of the user and the type of software in question.

2.3. Electronic Design Automation

Electronic Design Automation (EDA or ECAD) is a category of CAD tools for designing electronic systems such as printed circuit boards and integrated circuits.

Before EDA, integrated circuits were designed by hand and manually laid out. The earliest EDA tools were produced academically. By the mid-70's the first EDA tools for placement and routing were developed. One of the most famous was the "*Berkeley VLSI Tools Tarball*", a set of UNIX utilities used to design early VLSI systems. The beginning of industrial EDA was at 1981, as the larger electronic companies pursued EDA manually until then. Now EDA tools work together in a design flow that designers use to design and analyze entire semiconductor chips.

EDA led to the development, massive production and cost reduction of high-tech conveniences such as cell phones, navigation systems, media players etc. Nowadays EDA has an extraordinary effect on human life, as almost everything and every daily task have been influenced by this. The progression of microprocessor technology in terms of performance and features made the computer an essential tool and part of everyday life.

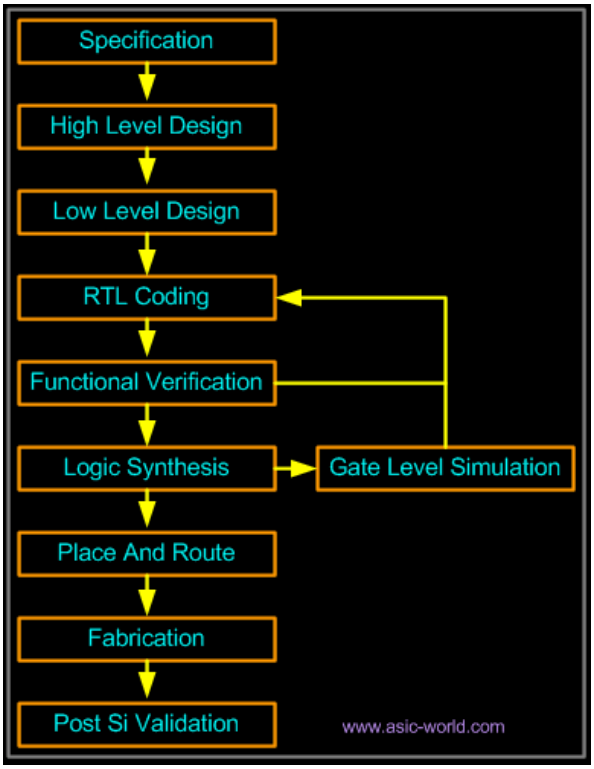
EDA has increased importance in the latest years, with the continuous scaling in semiconductor technology, because with the DSM era there are a lot of problems to be faced. Some of them are Design for Testability (DFT) and Automatic Test Pattern Generation (ATPG), lithography etc. Moreover, the evolution of the tools is necessary in order to overcome the difficulties of this era.

2.4. Design Flow

Design flows are the explicit combination of EDA tools to accomplish the design of an integrated circuit. Moore's Law has driven the entire IC implementation RTL to GDSII design flows from one which uses primarily standalone synthesis, placement, and routing algorithms to an integrated construction and analysis flows for design closure. [wikipedia]

Due to the progress being made by the semiconductor industries, e.g. the transistors' scaling, reducing the interconnection delay has become the great challenge. This fact led to a new way of thinking about integrating design closure tools and new scaling challenges for the current state of the art tools uprised, such as leakage power, variability, and reliability. There are two discrete flows for ASIC and FPGA designs. This thesis is based on the ASIC flow.

As mentioned above all EDA tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips. In this section is described a typical design flow, as there are some variations, e.g. for low-power design. The picture 2, below, illustrates the ASIC flow.



Picture 2: Front-end and Back-end flow(asic-world.com)

The whole process it can be separated in 2 domains:

- Front-end flow
- Back-end flow

2.4.1. Front-end flow

The Front-end flow is the process that guides from the concept to the netlist of logic-gates of a circuit. It includes steps, such as architectural design, simulation and synthesis. The front-end flow finishes at the Logic Synthesis step as depicted in the picture 2.

- Specification: The step at which are described important parameters of the design, e.g. what the design should do.
- High-level design: Various blocks are defined and description of the communication between them. Description is given in high-level languages (SystemC, C, C++).
- Low-level design: It is described how each block is implemented. It contains details about FSMs, counters, registers etc.
- RTL coding: The step at which Low-Level design is converted into Verilog / VHDL code, using synthesizable constructs of the language.
- Functional Verification: It is verified that the design does its expected function. Testbenches are created to apply all possible stimuli at the input and check the output.
- Logic Synthesis: Is the process in which synthesis tools take RTL code, target technology and constraints as inputs and maps the RTL to target technology primitives. After the gate-level netlist is created, timing analysis is done to check that the mapped design is meeting timing requirements.
- Gate-level Simulation: Check if the Design Under Test (DUT) is functionally correct.

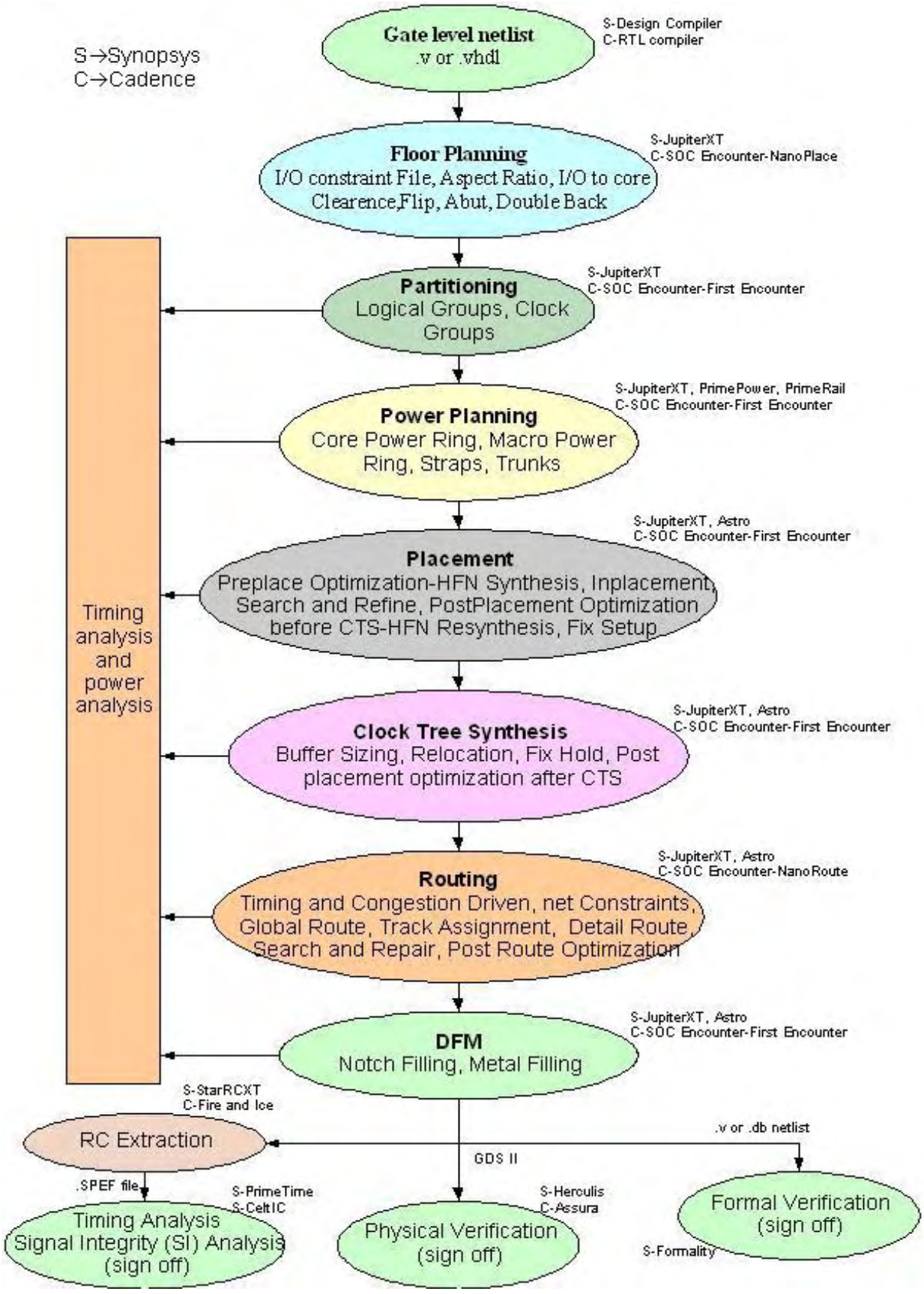
Before passing the netlist to the back-end flow, usually it is done Formal verification and insertion of scan-chains.

- Formal verification: Check if the RTL to gate mapping is correct
- Scan-chain insertion: Insert scan-chain in case of ASIC for design-for-testability(DFT) [asic-world]

2.4.2. Back-end flow

Back-end flow or physical implementation is the step in the standard design cycle which follows after the Front-end. At this step, circuit representations of the components (devices and interconnects) of the design are arranged on a piece of semiconductor material. More specifically they are converted into geometric representations of shapes which, when manufactured in the corresponding layers of materials, will ensure the required functionality of the design. The next step after Physical Design is the Manufacturing process or Fabrication Process that is done in the Wafer Fabrication Houses.

The main steps of the back-end are described in the picture 3.



Picture 3: Detailed back-end design flow with EDA tools and file format

Gate-level netlist: Is the circuit's synthesized netlist, produced after the completion of the front-end flow. It includes only standard-cells and their interconnections, as well and primary inputs and outputs of the circuit.

Floorplanning: Is the process in which the area of the design, the IO structure and the aspect ratio are decided. The usual process is to find structures that should be placed close together, and allocates space for in order to meet the, sometimes, conflicting goals of available space (cost of the chip) and the required performance. During this process some components such as the macro's used in the design, memory, other IP cores and their placement needs, the routing possibilities and also the area of the entire design (core area/die area), are taken into account in order to find the most suitable place for them, as these components can have a dramatic effect on the performance of the chip.

Partitioning: Is a process of dividing the chip into small blocks. This is done mainly to separate different functional blocks and also to make placement and routing easier. Partitioning can be done in the RTL design phase when the design engineer partitions the entire design into sub-blocks and then proceeds to design each module. These modules are linked together in the main module called the TOP LEVEL module.[wiki]

Placement: Is the process of placing the modules of the design, described in the gate-level netlist, in the core area decided in the floorplan step.

Clock-Tree Synthesis (CTS): Before CTS, clock is not propagated and considered ideal. Clock tree begins at source clock and ends at pins of a flop.

Routing: There are two types of routing in the physical design process, global routing and detailed routing. Global routing allocates routing resources that are used for connections. Detailed routing assigns routes to specific metal layers and routing tracks within the global routing resources.[wiki]

Signoff: Checks the correctness of the layout design, before it can be taped-out. There are several categories of signoff checks:

- *DRC* - Also known as geometric verification, this involves verifying if the design can be reliably manufactured given current photolithography limitations. In advanced process nodes, Design-for-Manufacture (DFM) rules are upgraded from optional (for better yield) to required.
- *LVS* - Also known as schematic verification, this is used to verify that the placement and routing of the standard-cells in the design has not altered the functionality of the constructed circuit.
- *Formal Verification* - The logical functionality of the post-layout netlist is verified against the pre-layout, post-synthesis netlist.

- *Voltage-drop analysis* - Also known as IR-drop analysis, verifies if the power-grid is strong enough to ensure that the voltage representing the binary high value never dips lower than a set margin.
- *Signal-integrity analysis* - Noise due to crosstalk and other issues is analyzed, and its effect on circuit functionality is checked.
- *Static-timing analysis (STA)* - Is used to verify if all the logic data paths in the design can work at the intended clock-frequency.
- *Electromigration lifetime checks* - To ensure a minimum lifetime of operation at the intended clock frequency without the circuit succumbing to electromigration.
- Once the design has been physically verified, optical-lithography masks are generated for manufacturing. The layout is represented in the GDSII stream format that is sent to a semiconductor fabrication plant (fab).

3. Placement

In order to produce an ASIC, a robust architecture of sub-circuits that meets the specifications must be designed and standard techniques for transporting an architectural design to a physical design with timing, area and power limitations. The development of digital circuits must depend on a detailed design due to their incremental complexity and large cost of their creation of a lithographic photo-mask.

Physical implementation starts given the physical design's behavioral description language.

Then the structure of a circuit must be specified in a structural **R**egister **T**ransfer **L**evel (RTL) language, a high-level hardware description language (HDL) for defining digital circuits. RTL specifications are turned into gate-level netlists, and the circuits are described as a collection of registers, Boolean equations, control logic such as "if-then-else" statements as well as complex event sequences The most popular RTL languages are VHDL and Verilog.

The circuit description does not contain the exact coordinates of every gate in the specified circuit area and does not contain the exact style of interconnections neither, so the next step is to determinate the placement decision of where to place all electronic components, circuitry, and logic elements in a generally limited amount of space and routing of the circuit, which decides the exact design of all the wires needed to connect the placed components. These two operations belong to the same step because the routing follows the placement description and builds up to it, in the way that a good placement can lead to an easier routing decision. Both of these operations are NP-hard problems, so there is a try to find approximate algorithms based on heuristics or branch and bound strategies.

With the placement and routing step, the circuit is fully described. Therefore the characteristics of the design can be computed. The final step is to perform signoff, in order to test all the signoff constraints described in chapter 2.

3.1 The Placement problem formulation

The general purpose of solving the VLSI placement problem is to find the best position of each module of a circuit on a specific area with the respect to a cost function that must be minimized.
More specifically, given:

- A netlist of cells from a pre-defined semiconductor library
- A mathematical expression of that netlist as a vertex, edgeweighted graph
- Constraints on pin-locations expressed as constraints on vertex locations / aspect ratio that the placement needs to fit into

- One or more of the following: chip-level timing constraints, a list of critical nets, chip-level power constraints

Placement process returns:

- Cell/vertex locations to minimize placement objective subject to constraints
- No two cells/vertexes overlap
- Placement is routable

An important part of the placement problem is to understand the critical cost functions that must be minimized. Some of these functions might be:

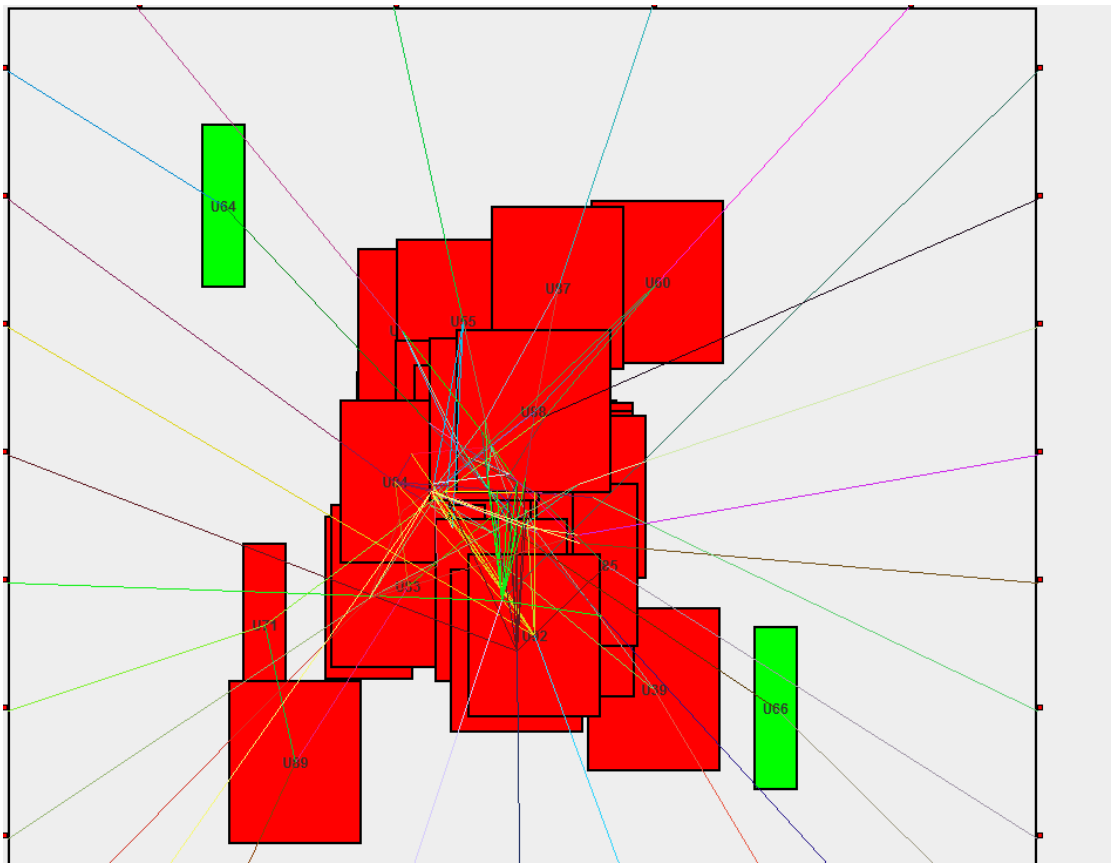
- Minimization of the expected length of longest wire, or the sum of the length of all the wires in the design, which is the primary objective of most existing placers. This not only helps minimize chip size, and hence cost, but also minimizes power and delay, which are proportional to the wirelength.
- Minimization of wire routing congestion. While it is necessary to minimize the total wirelength to meet the total routing resources, it is also necessary to meet the routing resources within various local regions of the chip's core area. A congested region might lead to excessive routing detours, or make it impossible to complete all routes.
- Minimization of timing delay. The clock cycle of a chip is determined by the delay of its longest path, usually referred to as the critical path. Given a performance specification, a placer must ensure that no path exists with delay exceeding the maximum specified delay.
- Minimization of power. Power minimization typically involves distributing the locations of cell components so as to reduce the overall power consumption, alleviate hot spots, and smooth temperature gradients.
- The minimization of the core area, die area, and other area constraints of the design, which come up for different reasons (remote devices, economical reasons etc.).
- The minimization of the distance of modules that use the same clock.
- A secondary objective is placement runtime minimization.

The optimal is to minimize every function but this is impossible because some functions have conflicts thus when one minimizes a variable the other may maximizes its own variable, for example when timing delay is minimized the power is grown. A good placement analysis must respect the cost functions that must be optimized and find a close to the golden mean that keep the balance between the critical variables. Another issue with the cost functions is that the evolution of technology changes their importance in the export of results. For instance, before the 90 nm-node transistors the basic timing delay was existed by a bad minimization of a specific area, but now it is proved that total delay is determined in a very grate percentage by routing, and the cost function that matters more is the total wire-length minimization.

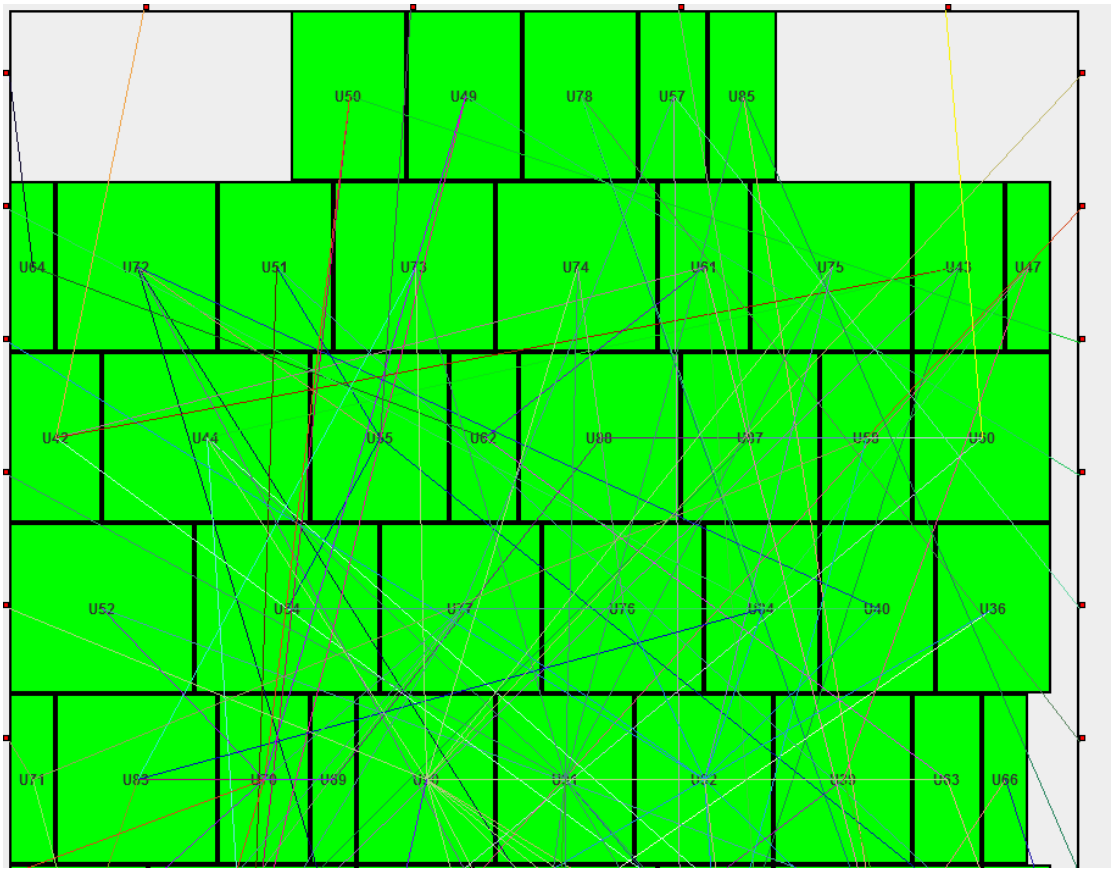
Placement problem is a NP-hard problem, which means that it cannot have a solution in polynomial upper time limit. If can be proved that the decision of a placement is a NP-complete problem then can be proved that the placement problem is an NP-hard problem. Below there is an explanation, but not the entire mathematical proof, as it is out of scope of this thesis:
The placement decision problem can be defined as the algorithm that decides given some modules, pins, nets, a specified area, if all modules can be positioned correctly and legally in the placement area. Of course it cannot be decided in polynomial but in exponential time because every combination of two cells must be checked and that leads to a quadratic problem. Officially the placement decision problem can be retreaded to the subset sum problem, which decides given a set of integers, if there is a non-empty subset whose sum is exactly zero, and is a NP-complete problem. Therefore the decision of a placement is a NP-complete problem.
Also it is proved that if an optimization problem H has an NP-complete decision version *L*, then *H* is NP-hard, thus the placement problem is NP-hard because it optimizes a cost function.

3.2 Classification of placement algorithms

The placement problem can be divided in two steps:
Global placement: It has also mentioned as relative placement, because it uses constructive or iterative algorithms in order to find a near-optimal placement, as a fist attempt of placing the cells into the placement area, usually by dividing it into core bins. This placement has overlaps between cells.
Final placement: It has also mentioned as detailed placement. Using the global placement as an input and a local final placement algorithm, it cuts the core area into core bins and finds a near-optimal placement solution for every core bin. Usually the output is a legal placement proposition, but the inner-solutions are not necessarily legal. If the output has overlaps then a legalization step is necessary to export a result that can be constructed in the psychical design. Additionally, in many final placement algorithms there is a last try to improve further a legal solution, calling post placement optimization. Wire or cell positions can be improved in this step, depend on the cost function, usually by swapping cells or wires movement. A global post placement optimization is usually done as a separate design flow step, after placement completion, which can add buffers in the design or use biggest/shorter standard cells to improve placement's objective subjects. No further discussion will be done for this last step because it is out of this thesis scope.



Picture 4: Global placement of a circuit



Picture 5: Detailed placement and legalization for the same circuit

Placement algorithms can also be split into constructive and iterative.

Constructive placement improvement algorithms deal with every cell once, but iterative algorithms have many reiterations of a curtain procedure and deal with every cell iteratively. Constructive algorithms are usually very fast and produce good results because of their global view of the problem. However, they are generally restricted in the choice of objectives and often do not yield the global optimum of the placement problem because of sacking in local minimums.

Iterative placement improvement algorithms aim at improving existing solutions, especially initial placements usually obtained with constructive algorithms. Typically, in one iterative step they select a small and local subproblem to be solved by exact or heuristic methods. These algorithms also divide into two classes depending upon whether they apply random or deterministic techniques. Iterative improvement methods based on randomized algorithms never reject better solutions, but they also accept intermediate placements of inferior quality with low probabilities. Thus, they have the ability to escape local optima and to approach the global optimum arbitrarily close if sufficient computation time is provided. Since this is not always practicable, particularly for large circuits, layout quality is compromised.

The philosophy of output produced by an algorithm is another way of classifying the placement algorithms. Some algorithms generate the same solution when presented with the same problem, i.e., the solution produced is repeatable. These algorithms are called *deterministic* placement algorithms. Fixed connectivity rules, cause and effect algorithms, or simultaneous equations solution are deterministic and always produce the same result for a particular placement problem and nothing else could be produced. Some algorithms, on the other hand, work by randomly examining configurations and may produce a different result each time they are presented with the same problem. Such algorithms are called as *probabilistic* placement algorithms.

Placement algorithms can also be classified from their solution approach of netlength minimisation, to linear, quadratic or neither of the two. But to understand the difference first there must be defined some netlength metrics that the algorithms deal with, in the next paragraph.

3.3 Net metrics

After implementing placement process in a design, cells obviously have standard coordinates inside the core area. This means that a close to absolute total wire length can be calculated, summarizing every net. Because of not having the exact net distance of nets before routing stage, there is a need to define some netlength metrics with which the wirelength can be calculated and then optimized. Before proceeding,

consider that in all distance metrics, cells are defined as points, with coordinates their lower left point or their center.

3.3.1 Lk-norm induced distance metric

$$dk(p,q) = \sqrt[k]{|p_x - q_x|^k + |p_y - q_y|^k}, \quad p,q \in R^2, \quad k \in \aleph$$

Let p and q be two modules, and px py, qx qy their coordinates in x and y axis.
L1 is the Manhattan distance and L2 is the Euclidean distance which is mostly using as a netlength distance metric. Also, when $k \to \infty$ it gives:

$$d_q(p,q) = \lim_{k\to\infty} d_k(p,q) = \max(\,|p_x - q_x|, |\,p_y - q_y|\,)$$

3.3.2 Rectilinear Steiner Tree and Minimum Spanning Tree

Given *n* points in the plane, it is required to interconnect them all by a shortest network which consists only of vertical and horizontal line segments. It can be shown that such a network is a tree whose vertices are the input points plus some extra points (Steiner points). [wikipedia]

Assuming that until nowadays only horizontal and vertical wires are used, the total minimal netlength is achieved by connecting pins in each net with a Steiner tree such that there is no intersect between trees. Although the routing of a whole net with multiple nodes is better represented by the rectilinear Steiner tree and it can be considered as the smallest wirelength measure, the rectilinear minimum spanning tree (RMST) provides a reasonable approximation and wire length estimate. rectilinear minimum spanning tree can be considered as a RSTM without extra (Steiner) points, so every node is interconnected in the shortest network which consists only of vertical and horizontal line segments. For the same number of nodes with specific coordinates, Hwang proved that:

$$RMSTlength \leq \frac{3}{2}\, RSMTlength$$

The RSMT is an NP-hard problem but proof is out of the scope of this thesis.

3.3.3 Clique

The clique model is very popular in many VLSI problems. This model is based on proper conversion of nets into clique subgarphs. The given weight for every edge that builds the clique is computed by the following type:

$$w(i,j) = \frac{2}{n}$$

Where i, j the nodes connected by an edge and k the total number of the cliques' nodes. This type ensures that the effect of the total weight of big cliques is relatively decreased against small cliques that are usually the majority in physical designs. On the other hand this type ensures that the total weight effect is balanced to the size of every clique.

Then, the length of a net $n \in N$ is the total distance of pairs of pins in the net:

$$CL_k(n) = \frac{1}{(|n|-1)} \sum_{p \in n} \sum_{q \in n} d_k(A(p), A(q))^k$$

The clique netlength may be evaluated in time $O(|n|^2)$. For an improved approximation of the linear net model by the quadratic one, Gordian-L(Sigl, Doll and Johannes [1991]) use the following iterative net model: in iteration k = 2… one optimizes essentially:

$$GordianL_x{}^{(k)}(n) = \frac{1}{|n|-1} \sum_{p,q \in n} \frac{(x_k(p) - x_k(q))^2}{|x_{k-1}(p) - x_{k-1}(q)|}$$

and $L^{(k)}(n) = GordianL_x{}^{(k)}(n) + GordianL_y{}^{(k)}(n)$.

Recall that linear clique was the model of choice for a-priori topologies, so in the limit one can obtain the best achievable result. But after all, one cannot expect more than linear convergence of this method, so in general this approximation method will be too slow.

3.3.4 Star

This model is similar to clique model, where the hyperedges of the graph are converted to star subgraphs.

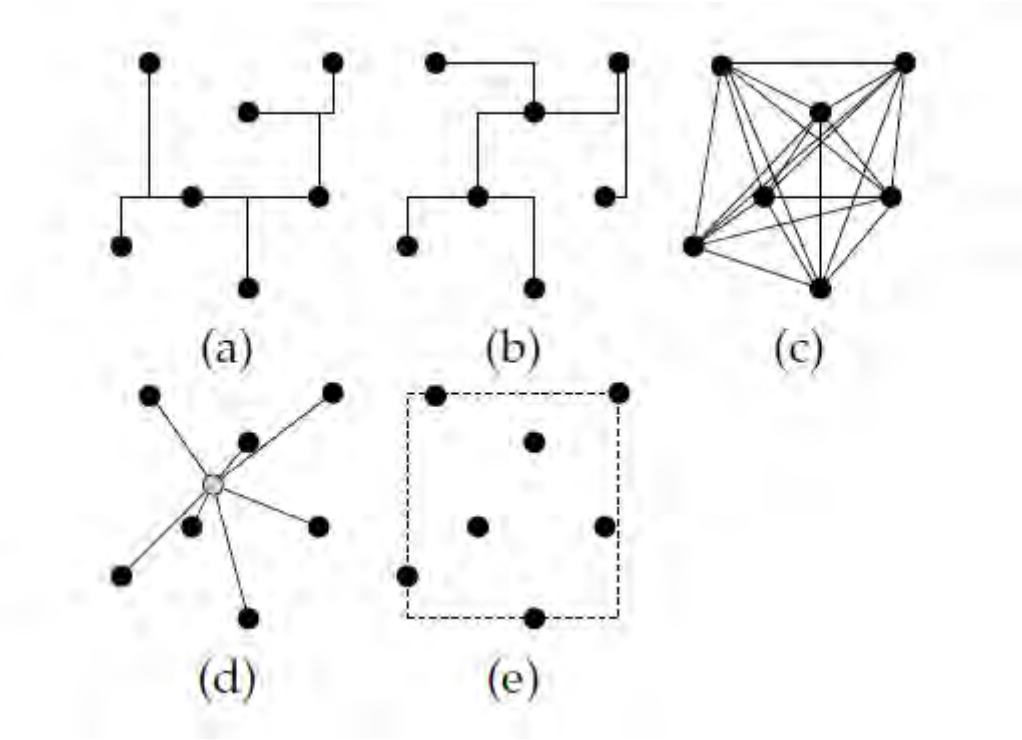
3.3.5 Bounding Box

The bounding box is by far the simplest net metric, as it defines that that total netlength of a circuit is the summary of the half perimeter of the bounding box surrounding every net.

$$BB(n) = \max_{p \in n}\left(A_x(p)\right) - \min_{p \in n}\left(A_x(p)\right) + \max_{p \in n}\left(A_y(p)\right) - \min_{p \in n}\left(A_{xy}(p)\right)$$

Complexity of calculating the total netlength using the bounding box metric is $O(n)$.

Now considering and comparing the complexity of bounding box metric and the clique metric, it is obvious that algorithms that use bounding box metric called linear minimization algorithms, and algorithms that use the clique metric called quadratic minimization algorithms. The figures below can show the graph style of every metric and the graph style of the difference between linear and quadratic calculation.



Picture 6: Rectilinear Steiner Tree and Minimum Spanning Tree(a,b), clique(c), star(d), bounding box(e) [3]

3.4 Minimizing Quadratic Netlengths

Both quadratic clique and star netlengths can be expressed by matrix notation. If the corresponding problem is relaxed by removing the no-overlap constraints the quadratic netlength we can be minimized many methods, with the most popular to be the Conjugate Gradient Method.

The Conjugate Gradient Method works on parabolic functions by taking steps towards the global minimum. The method requires that the quadratic matrix is symmetric and

positive. In each step of this iterative procedure the current solution is moved in the direction of the eigenvalues of the matrix. When the solution is deemed close to minimum the procedure ends. The complexity of this method depends on the spectral condition number k of the matrix and it is $O(\sqrt{km})$ when m is the number of non-zero elements of the matrix and sparse matrix data structures are used.

The matrix is in general sparse. Each row corresponds to a module. For the star model the number of non-zero elements in a row is roughly equal to the number of nets the module of that row is connected to. For the clique model this number is equal to the number of modules the module is connected to. It is generally assumed that the average number of nets each module is connected to is usually less than 5.

3.4.1 Preconditioning

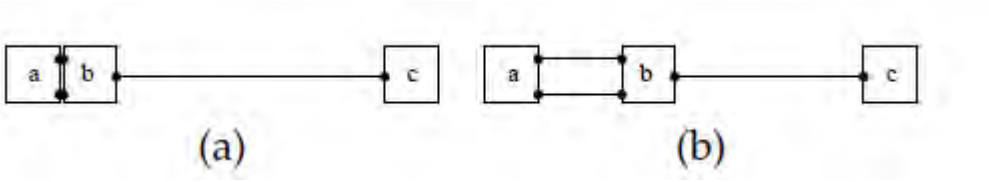
As mentioned above the Conjugate Gradient Method depends on the spectral condition number k. This can be improved by using a preconditioner. The essence of the preconditioning method is that instead of solving a system of the form $Ax = b$ one can solve an auxiliary system $M^{-1}Ax = M^{-1}b$. If $M^{-1}A$ is better conditioned than A we may get faster convergence. The matrix M is called a preconditioner and the problem is of course determining M such that we can easily find M^{-1} . Ideally one would pick A but since there is no information about A^{-1}, an alternative choise has to be taken. The simplest preconditioner is the diagonal of A and the most popular is an incomplete cholesky factorization of A. We will not discuss preconditioning further here but even diagonal preconditioning can improve the order of convergence significantly.

3.5 Minimizing Linear Netlength

The unconstrained linear bounding-box formulation can also be solved in efficient time. Here the objective function is the dual of a minimum-cost-flow problem. The dual problem can be solved and by using LP-duality a solution to the primal problem can be determined, as it proposed by Weis and Mlynski. Nevertheless even the fastest algorithms of this kind have quandratic running time, $O\left(mnlog\left(\frac{n^2}{m}\right)\right)$.

Linear net models, approach better the actual post-route wirelength, but lead to various drawbacks during placement. The first reason is the optimization issue as

linear functions are not differentiable. Despite the fact that placement minimizing linear bounding-box netlength can be considered as the dual of a MinCostFlow problem and computed exactly, linear solutions stuck in local minima and it is difficult to compute global optima in acceptable time if the number of variables is in the millions. The second reason is that placements optimizing linear netlength are not unique in general. Those computed by such a MinCostFlow approach tend to implode cells in some particular corner of the core area and do show significant overlaps, which are solved with many arbitraries and cause worst wirelength solutions as a result. Such an effect does not show up for strictly convex net models. Nevertheless, many non-linear methods exist to approximate the linear netlength model, also called analytic. These approaches are introduced in next paragraphs.



Picture 7: Linear (a) and quadratic (b) solution for wirelength minimization [3]

4. Placement methods and Algorithms

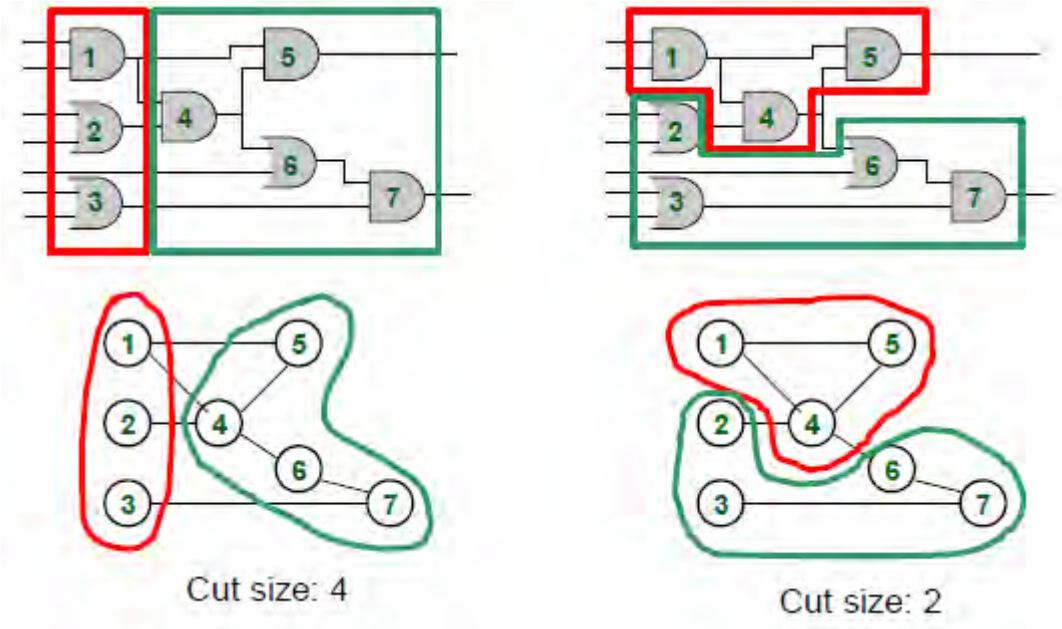
As mentioned in the previous chapter, traditionally, placement is separated in two stages, global placement and final placement. In this chapter are presented, analytically, the methods of these two stages and the most known algorithms.

4.1 Global placement Methods

The main purpose of global placement is to distribute the cells evenly over the placement region and optimize certain objectives e.g. wirelength, timing, power etc. It pays attention to the relative positions of the cells globally.

4.1.1 Graph partitioning

The most used technique for global placement is based on graph partitioning. Given a gate-level circuit, the goal of circuit partitioning problem is to divide the circuit into *K* roughly equal-sized partitions. The main goal of this method is to minimize the number of hyperedges that connect nodes from one partition to another, which is typically called the "cutsize" in the literature. Other objectives include critical path delay, total power consumption, etc. Due to these objectives, partitioning algorithms can be divided into linear and analytic, that use linear solutions for the "cutsize" and quadratic approaches to compute other objectives. These two categories are analyzed further below:



Picture 8: Cut-size example, institute of Microelectronics Systems (MES)

4.1.1.1 Linear based partitioning heuristics

Every module that has to be placed is defined as a node of a graph with specific weight. Weight of nodes is computed usually by the cell's fanout and its area or other critical objectives. In this iterative method, each bisection has to be chosen such that the weight of nodes on either side is close to equal while at the same time the number of edges cut by the bisection is kept at a minimum. After the components are pre-placed with an initial placement algorithm, partitioning starts with the hole graph as the first partition. In every step a partition with total weight W is cut in two other with weight W/2, and this is done recursively for every new partition until the number of edges cut by the bisection is the minimum. With this technique it is succeeds to keep close to each other all the highly connected components.
Linear graph partitioning approaches do not accomplish wirelength minimization directly but in every, which can achieve placements inside a partition that is faster and dealing with a smaller number of components that are highly connected. If the Simulated Annealing approach will be combined with the graph partitioning technique, it will accept any kind of objective, in particular wirelength, but it can be applied only for tiny instances due to its huge complexity, so for large scale circuits another technique has to be used. Analytic approaches can cover this drawback, especially in global placement, as in detailed placement the core area can be partitioned and can be handled separately. Nevertheless this global placement problem is NP-hard (reduce from 2-partition) because of the balance requirement.
Two famous linear graph-partitioning algorithms are of Kernighan-Lin[1979] and Fiduccia-Mattheyses[1982].

Kernighan-Lin: Given a netlist, the first step is to create an edge-weighted undirected graph *G* that represents the circuit. We typically use so called the *k*-clique model, where a net that contains *k* gates forms a *k*-clique in *G*, and each edge in the clique gets a weight of $1/(k-1)$. In case an edge (x, y) already exists from a prior net conversion, it weight has to be updated properly. The KL algorithm is applied on this graph, so the cutsize and gain are computed based on it, instead of the original circuit. Next, an initial balanced bipartitioning solution ($P1$, $P2$) of G is acquired, usually randomly. For a cell $x \in P1$, the external cost of x is defined as follows:

$$E_x = \sum_{i \in P_2} c(x,i)$$

where $c(x, i)$ is the weight of the edge $e(x, i)$. This Ex is the expression for the sum of the weight of edges which connect x with nodes that are in other partition, where the neighbors of x are defined to be the nodes that are connected with x via an edge. Sequentially the internal cost of x is defined as follows:

$$I_x = \sum_{i \in P_1} c(x,i)$$

This *Ix* is the expression of the sum of the weight of edges that connect *x* and its neighbors in the same partition. Finally, the gain of swapping *x* and *y* is defined as follows:

$$gain(x, y) = (Ex - Ix) + (Ey - Iy) - 2c(x, y)$$

Before the first pass starts all cells are considered as locked. Once the pass begins, the following procedure is repeated at every swap until all cells are locked. Firstly, the gain of all unlocked pairs is computed and then the pair with the maximum gain is swapped and the cells in the pair are getting locked. In the third step, the gain and the current cutsize are recorded. When the pass is finished, the first *K* swaps with the biggest gain are accepted, leading to the minimum cutsize discovered during the entire pass. If the initial cutsize has reduced during the current pass, it means that a better solution hs been discovered, and another pass is tried on using as initial solution the best solution discovered from the current pass; otherwise the algorithm is terminated considering that it cannot find a better solution. Since the cells' swapping, the area is always balanced between the two partitions. Note also that the entire Kernighan and Lin algorithm can be repeated with another random initial solution.

```
Kernighan_Lin (A, B)

compute D values
for i from 1 to n
    Locked[ i ]:= false

BestCost := Cost[ 0 ] := cutsize(A, B)
BestChange := 0
for s from 1 to n/2 do
    Cost[ s ] = ∞
    for i, j from 1 to n such that v_i ∈ A and Locked[ I ] = false and v_j ∈ B and Locked[ j ] = false do
        if 2 ω[i, j] – D[ i ] – D[ j ] < Cost[ s ] then
                Pair[ s ]   := ( i, j )
                Costs[ s ] := 2ω[ i, j ] – D[ i ] – D[ j ]
    (imin, jmim) := Pair[ s ]
    Locked[ imin ] = Locked[ jmin ] = true
    for i from 1 to n such that Locked[ i ] = false do
        if v_i ∈ A then
                D[ i ] = D[ i ] – 2ω[ i, jmin ] + 2ω[ i, imin ]
        else
                D[ i ] = D[ i ] – 2ω[ i, imin ] + 2ω[ i, jmin ]
    Cost[ s ] := Cost[ s – 1 ] + Cost[ s ]
    if Cost[ s ] < BestCost then
        BestChange := s
        BestCost := Cost[ s ]

    for s from 1 to BestChange do
        exchangePair[ s ]
```

Picture 9: Kernighan-Lin algorithm, institute of Microelectronics Systems (MES)

Fiduccia-Mattheyses: The algorithm starts with an initial balanced bipartitioning solution (*P*1, *P*2) of the given hypergraph, which is usually obtained randomly.
For a cell $x \in P1$, $FS(x)$ is defined as the number of nets that have *x* as the only cell in *P*1.

$TE(x)$ is defined as the number of nets that contain x and are entirely located in $P1$, i.e., all cells in the net are partitioned in $P1$. Finally, the gain of moving x from $P1$ to $P2$ is simply:

$$gain(x) = FS(x) - TE(x)$$

Before the first pass starts all cells are considered as locked as in Kernighan and Lin algorithm, and the gain of all cells is computed based on the initial partitioning. Also the cells have to be added to the bucket structure. Once the pass begins four steps have to be repeated at every move until all cells are locked. Firstly, the cell with the maximum gain is considered as legal. A cell move is legal when while moving it to the other partition it does not violate the core area constraint. Secondly, the chosen cell is moved and locked it in the targeted partition. After these steps, the gain values of the moved cell's neighbors their positions in the bucket are updated, and finally the gain and the current cutsize is recorded. At the end of the pass, the first K moves are identified, accepted and lead to the minimum cutsize during the entire pass. If the initial cutsize has reduced during the current pass, we attempt another pass using the best solution discovered from the current pass as the initial solution; otherwise the algorithm is terminated. Note that the entire FM algorithm can be repeated with another random initial solution.

Step 0: Compute the balance criterion
Step 1: Compute the cell gain Δg_1 of each cell
Step 2: $i = 1$
- Choose base cell c_1 that has maximal gain Δg_1 , move this cell

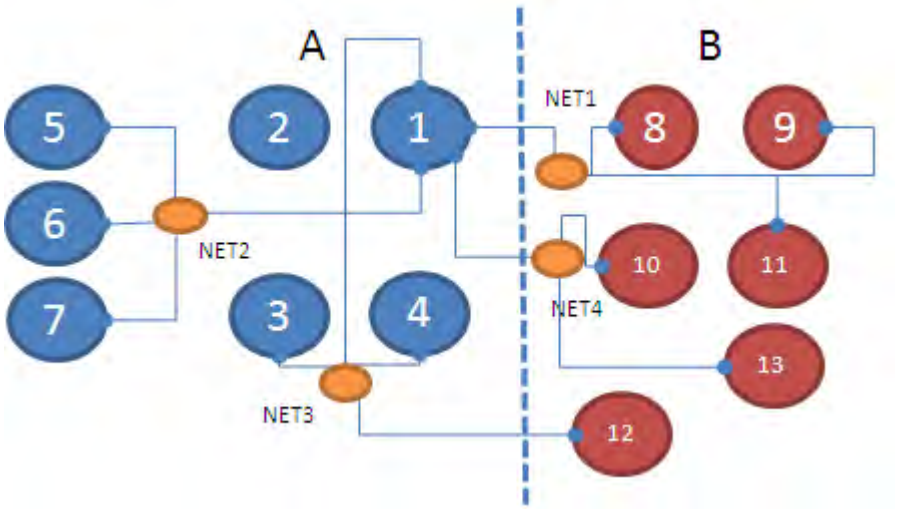
Step 3:
- Fix the base cell c_i
- Update all cells' gains that are connected to critical nets via the base cell c_i

Step 4:
- If all cells are fixed, go to Step 5. If not:
- Choose next base cell c_i with maximal gain Δg_i and move this cell
- $i = i + 1$, go to Step 3

Step 5:
- Determine the best move sequence $c_1, c_2, .., c_m$ $(1 \le m \le i)$, so that $G_m = \sum_{i=1}^{m} \Delta g_i$ is maximized
- If $G_m > 0$, go to Step 6. Otherwise, END

Step 6:

Picture 10: Fiduccia-Matheyses algorithm



Picture 11: Fiduccia-Matheyses example (wikipedia)

4.1.1.2 Analytic based graph partitioning heuristics

The analytic partitioning heuristics are closely related to the graph-partitioning heuristics. But instead of minimizing cuts they use the analytic placement to guide the position of the cut-lines. The oldest analytic method found in the literature is from 1970 and by Hall, and the most popular algorithms are of Hagen and Kang (EIG), and of Yang and Wong (FBB).

EIG: this algorithm utilizes the second smallest eigenvalue and its eigenvector of a netlist matrix to optimize so called the "ratio cut" metric. This ratio cut metric, defined as $c(X, Y)/|X||Y|$, where $c(X, Y)$ denotes the cutsize between the two partitions X and Y, managing to minimize the cutsize and balancing the area of the existing partitions. Given an undirected graph that represents the connectivity among the nodes, and its Laplacian matrix, the eigenvector of the second smallest eigenvalue of the matrix defines a one-dimensional placement of the graph's nodes [Hall 1970]. In this placement solution, the quadratic length of every edge in the graph is minimized under the constraint $\sum_i xi = 1$, where xi is the x coordinate of node i. It is also shown that the second smallest eigenvalue of the matrix is the lowest bound of the ratio-cut metric. According this, their analytic partitioning algorithm computes bipartitioning solutions with minimum ratio-cut metric, using the one-dimensional placement. So, given a gate-level circuit, first its undirected graph representation must be derived, based on the standard k-clique model. In this model, a net with k gates forms a k-clique, and each edge in the clique gets a weight of $1/(k - 1)$. The remaining part of the algorithm proceeds as follows:

1. Building of the $n \times n$ Laplacian matrix $Q = D - A$, where n is the number of nodes in graph G. A is the *adjacency matrix*, where each entry aij denotes the weight of edge $e(i, j)$ in G. D is the *degree matrix*, where each entry dii is the sum of the weights of all edges incident to node i in G.

2. Computation of the second smallest eigenvalue and its eigenvector of *Q* using Lanczos method.
3. Sort of the nodes in *G* based on their values in the eigenvector and obtain the node ordering $Z = \{v1, v2, \cdots, vn\}$.
4. Usage of *Z* to derive and evaluate $n - 1$ partitioning solutions. More specifically, we first obtain a bipartitioning solution ($\{v1\}, \{v2, \cdots, vn\}$) and compute its ratio cut metric.4 Next, we evaluate the ratio cut metric of ($\{v1, v2\}, \{v3, \cdots, vn\}$) from *H*, etc. Lastly, we choose the partitioning solution with the minimum ratio cut metric.

FBB: FBB performs flow-based bipartitioning. Given a flow graph *G* and a pair of source and sink nodes (*s*, *t*), the Maximum FlowMinimum Cut Theorem [Ford and Fulkerson, 1962] states that the maximum flow from *s* to *t* defines a bipartitioning of *G*, and the weight of the cut is minimized among all cuts separating *s* and *t*. Yang and Wong proposed an iterative method to repeat max-flow computations to find balanced solutions for the partition cuts from s to t, with complexity that approach asymptotically the same as a single max-flow computation, recycling the growing paths from the previous iterations. When a circuit has multi-terminal nets, they proposed a way to transform them into a flow network *G* so that any cut in *G* preserves the correct cutsize information in the circuit. Lastly, they proposed an effective way to achieve the next cut if the current cut is not balanced, which is based on making a minor perturbation to the current flow network.

More specifically, given a circuit *H*, the flow network G must be built, where each net $n = \{v1, v2, \dots, vk\} \in H$ is transformed as follows:
Nodes $v1, v2, \dots, vk$ were added into *G* if not added yet. Two auxiliary nodes *n*1 and *n*2 were added into *G* and are connected with bridging edge *e*(*n*1, *n*2) with its capacity value set to 1. Node *n1* is connected with *v*1, *v*2, ... , *vk* with edges of ∞ capacity. Node *n*2 is connected to *v*1, *v*2, ... , *vk* with edges of ∞capacity. Next, a pair of source and sink (*s*, *t*) has to be chosen randomly. After these, the fllowing must me repeated until a balanced bipartitioning solution is found:

The maximum flow from *s* to *t* using the growing path method must be found.
Then we construct the partitioning solutions based on the max-flow computation, which is done by cutting some subset of the saturated nets. Simultaneously, the best solution for the area balancing *C*(*X*,*X'*) must be found. Note that all of these solutions have the same cut-size, which is equal to the max-flow value. If a solution that satisfies the area constraint is found, the algorithm is terminated.
Nevertheless, the solution is not necessarily area-balanced. In this case must be examined if the area of *X* is smaller than the area lowest bound when a node $v \in Y$ that is contained in a cut net is chosen. Lastly, all nodes in *X* and *v* are merged into a single node, which becomes a new source *s*, and the algorithm starts again from the first step. Otherwise, if the area of *X* is bigger than the area upper bound, a node

$v \in X$ that is contained in a cut net must be found, all nodes in *X'* and *v* are merged into a single node, which becomes a new sink *t* and the algorithm starts again from the first step.

When the algorithm is in step 1 of some iteration, some of the growing paths are already found in previous iterations. This slows down the process' complexity by finding only the new additional growing paths, in order to obtain a new max-flow. Also, we choose *v* among the nodes in the cut net randomly, while we are finding a node *v* to merge with *X* or with *X'*. Thus, as the algorithm approaches closer to a balanced solution, we choose the best *v* among all the nodes in the cut nets.

4.1.2. Analytic and Relaxation Based Placement

The quadratic netlength formulations can be solved fast using numerical linear equations solvers, with the most popular the Conjugate Gradient Method, while the bounding-box formulation requires quadratic running time network-flow methods.

4.1.3 Force-Based Methods

The force-based methods can be defined as methods that handle modules as objects and nets as springs connecting the objects, and try to minimize the interconnection netlength by putting the "spring-system" into equilibrium, in respect to the Hooke's law. This kind of netlength minimization was introduced by Eisenmann and Johannes in 1998. The force-based method uses the analytic placement techniques to achieve an illegal overlapping placement which minimizes quadratic netlength. This heuristic proceeds iteratively introducing "repelling forces" between each module and bins on the placement area with overlap. In every iteration the current overlapping forces are added to the quadratic netlength and the combined quadratic function is minimized.

The force at a location $(x, y)^t$ is set to:

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

To improve the netlength, Eisenmann and Johannes also used a variation of the linearization method. Finally the Domino local search method which will be discussed later was used for final placement. Eisenmann and Johannes only used the heuristic on standard-cell circuits but Mo extended the force-based heuristic to macro-cells, with no linearization scheme to be considered. Also instead of introducing repulsive forces, a filling force is introduced between modules and empty regions of the placement area. Mo also considers orientation of the macro-cells. For each cell the new orientation is determined by considering all eight possible orientations and selecting the one which minimizes the external force. Since changing orientation of one cell will affect the force on other cells, care must be taken not to change orientation of too many cells at once.

Finally Routing and pad positioning are meditated. Routing is meditated by estimating congestion in each bin and pad positions are computing according to modules position. Specifically, the pads are included in the quadratic matrixes, but they are allowed to move across on dimension only. The iterative flow is divided into three stages. Stage one gives initial positions of the macro-cells. Stage two optimizes orientation and routing. Finally stage three removes any additional overlap.

Hu and Marek-Sadowska introduced a slight modification of Eisenmann and Johannes' method referred to as Fixed-point Addition and Relaxation (FAR). In their method they create an extra pseudo-module for each module and an extra pseudo-connection between each module and its associated pseudo-module. They noticed that fixed-points are easier to control than the repelling forces, and that their analytic placements can also be contained into a specific region, in comparison to the constant forces.

Attractor Repellor Approach (ARP) is another force-based method, proposed in 1999 by Etawil. It is a completely different method that the previous, because instead of looking at overlapping regions a repelling force is introduced between directly connected modules. This alternate idea may have many drawbacks, like too much overlap that grows the total wirelength in legalization stage. Therefore Etawil introduces attractive forces in low density regions.

4.1.4 Simulated Annealing for Global Placement

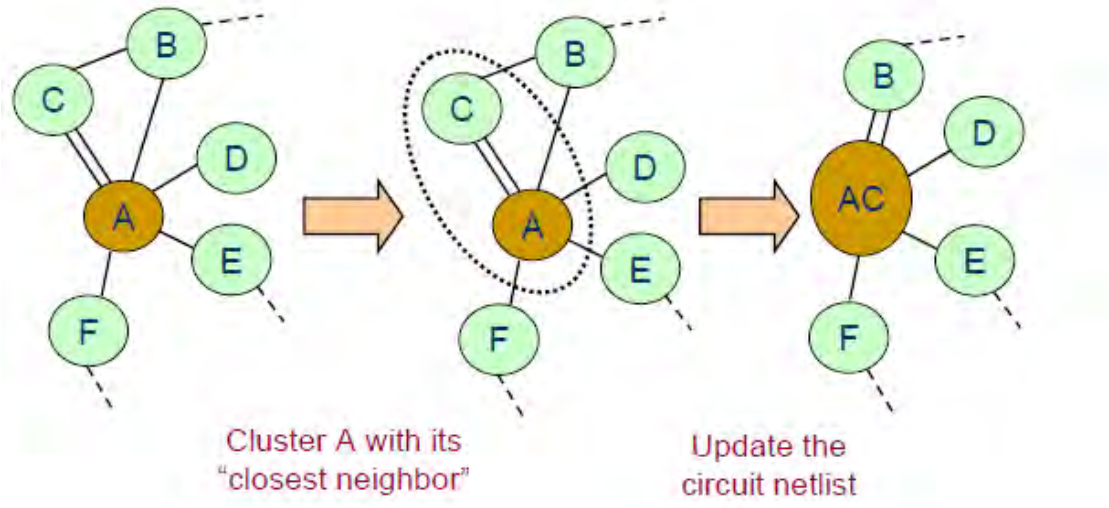
Simulated annealing (SA) is a generic probabilistic method for the global optimization problem. It firstly proposed from Kirkpatrick, Gelett and Vecchi(1983) and Cerny (1985). It results a good approximation to the global optimum of an objective function in a large, discrete search space with many local minima. In many cases Simulated annealing may be more efficient than any other heuristic.

Because of its simplicity, it has become the most famous meta-heuristic, used in many applications, including physical placement, if all else fails. However, simulated annealing applied to the placement problem has shown that it is high time costly when used for large scale circuits Nevertheless, it is used from the Dragon placer, for the global placement, and it is very used in final placement, even when it is used in partitions of the global placement.

4.1.5 Clustering

Some placement algorithms combine modules in clusters (clustering). Clustering is a powerful method dealing with sub-circuits, and can reduce the running time since there are fewer modules. Also, clustering can move a simple placement algorithm out of local minimum because it considers more than simple individual cell movement. Clustering methods have been used with the graph partitioning algorithms to improve

the speed of the partitioning. However clusters have also been used in detailed placement. Given a gate-level circuit, the goal of circuit clustering is to group gates and their interconnections into clusters and obtain the network of the clusters. Thus, after clustering, a cluster becomes a done of the graph, resulting reductions of the nodes and their interconnections. Sequentially, clustering is used before portioning and placement algorithms, to reduce of the run time complexity of any heuristic applied after it. Clustering objective functions is to minimize the connections from a cluster to another maximizing the number of connections inside a cluster, and to maximize the number of inter-cluster connections on any path. Typical constraints of this method may include the maximum cluster size and the maximum number of external connections for a cluster. The number of clusters to be obtained is not specified, and the area balance among the clusters is usually not required.



Picture 12: Clustering example

There are two ways to consider clustering problem in physical placement, in order to organize the clusters better for this specific problem:

- Produce the clusters strictly following the hypergraph representation of the circuit, as a general problem without considering placement.
- Produce the clusters based on placement heuristics and comute weights according to the objective function.

The most popular algorithms that use clustering based on placement heuristics are Rajaraman and Wong algorithm [Rajaraman and Wong, 1995], FlowMap algorithm [Cong and Ding, 1992] and Multi-level Coarsening algorithm [Karypis et al., 1997]. The first two are timing driven algorithms, where the longest path delay inside a cluster is minimized under a certain delay model. The last one minimizes the number of connections between clusters.

Rajaraman and Wong Algorithm: Given a directed acyclic graph which represents a gate-level circuit the algorithm produces a timing driven clustering solution.
In respect to the "general delay model" that uses, below are introduced the delays in the circuit:

- The delay inside each node of the cluster is computed.
- The inter-cluster edges have constant delay.
- The intra-cluster edges do not incur any delay.
- The size of each cluster is bounded by another constant.
- The critical paths delay from any primary input to primary output in the clustered network is minimized.
- Some nodes in the original DAG may be duplicated in the clustering solution.

The algorithm consists of two phases, labeling and clustering phase. During the labeling phase, the critical path delay from any primary input to each node is computed, adding both the node's delay and the delay from the primary input to the node (inter-cluster edge). During the labeling process, the clustering information about the nodes that will be clustered in the same cluster-node is also collected, and in case a node is included in multiple clusters, it is duplicated to exist in both clusters. More specifically, a $n \times n$ matrix D that contains all-pair critical path delays is created. Each entry at row x and column v, denoted by D(x, v), is the critical path delay from the output of x to the output of v in the DAG using node delay values only and ignoring all interconnect delays. After this, the labels of all primary input nodes are initialized to their delay values and all other nodes are set to zero. We then visit non-primary inputs in a topological order to compute their labels.
During the clustering phase, the nodes are visited in the opposite topological order, and the general clustering process takes place.

FlowMap Algorithm: This algorithm produces also a timing driven clustering solution, but in difference to the Rajaraman and Wong algorithm it uses the "unit delay model". In this model only the inter-cluster edges appear to have a unit delay while the nodes and intra-cluster edges do not incur any delay. The difference between the timing driven clustering using the unit delay model compared with the general delay model is that the number of external connections for each cluster is bounded by a constant. The maximum delay from any primary input to primary output in the clustered network is minimized in the solution. Some nodes in the original directed acyclic may be duplicated in the clustering solution.
FlowMap algorithm consists of two phases, the labeling and the mapping phase. During the labeling phase all nodes are met in topological order. For each node n the clustering set Cn is computed, as the set of nodes that will be clustered together in the same cluster. Another variable has to be computed, named label $l(n)$. This denotes the critical path delay from every primary input to v, where only the inter-cluster edges incur a unit delay. During the mapping phase, the nodes are visited in the opposite topological order, and the general clustering process takes place.

Multi-Level Coarsening Algorithm

Multi-level coarsening is an algorithm for clustering that first introduced as a part of hMetis algorithm for placement. hMetis algorithm released as the best bipaartitioning algorithm in Design Automation Conference (DAC) in 1997 [Karypis et al., 1997]. It consists of two phases, clustering and partitioning, to solve the balanced bipartitioning problem, where the given circuit is divided into two roughly equal sized partitions. This algorithm's target is to minimize interconnects from each cluster to another, respecting the "multi-level optimization" heuristic that obtains iterative clustering processes. During the first iteration, the highly connected nodes are grouped into level 1 clusters, and create a graph that contains them. This graph has less nodes than the original graph. Next level-1 clusters are grouped together to form level-2 clusters and their graph is created. This process is repeated until we group the clusters together in the target level *K*, in the clustering hierarchy, and the size of the graph is reduced spectacularly.

After the clustering, the level-*K* clusters have to be partitioned using any of the existing partitioning algorithms. After the partitioning process ends, every level-*K* cluster has to be decomposed to its level-*K*-1 clusters, and a partitioning algorithm is performed again to optimize further the current partitioning solution. This process of decomposing and optimizing the graphs' solutions is repeated until there are no clusters but the partitioned circuit. This clustering algorithm, because of considering its high level cluster interconnections, and because of its iterative nature, gives better partitioning solution.

More specifically, given a hypergraph that represents the original circuit, hMetis algorithm utilizes three algorithms to compute the multi-level cluster hierarchy, which are introduced below:

- Edge coarsening (EC): Initially, the nodes in the graph are unmarked and visited in a random order. Given an unmarked node v, the "neighbors" of v must be collected, which is the set of nodes that are unmarked and are included in the edges that contain v. For each neighbor n of v, the weight of edge (v, n) is computed by assigning a value 1/(|h| by1), where h denotes the edge that contains both n and v. After examining all neighbors of v, the neighbor with the maximum edge weight must be selected and v and n must be merged together. Both n and v so are marked that these nodes are not clustered again later. This process completes when all nodes are visited.
- Hyperedge coarsening (HEC): Initially, the nodes in the hypergraph are unmarked. If the hyperedges are not weighted they are sorted by size with an increasing order, otherwise are sorted by their weights decreasingly. In every case they are visited in the sorted order and there are examined if any of them contains any node that is already marked. If not, all nodes are grouped in the hyperedge to form a cluster. Otherwise, we skip to the next hyperedge. After visiting all hyperedges, each node that is not part of any cluster becomes a cluster of its own.
- Modified hyperedge coarsening (MHEC): This algorithm first applies hyperedge coarsing to the given hypergraph and after the hyperedges to be clustered have been selected, the algorithm visits them again in the sorted order. Then, for each hyperedge that has not yet been clustered there is examined if it contains any node that is unmarked, to cluster them all together. MHEC achieves reduce further the number of the inter-connections among clusters after balance the size among the clusters.

4.2 Final placement Methods

The aim of final placement subproblem, is to further improve the placement's solution quality. It is more constrained than the global placement as it optimizes the objectives and produces a legal placement. It usually uses more accurate models such as half-perimeter wirelength. Compared to legal placement there has been much less work in terms of final placement. At the following paragraphs there will be represented the most significant methods of final placement.

A classification of final placement methods is difficult because they may utilize a number of different algorithms within the same method, and some amount of legalization may be inherent in the proceeding global placement. There tends to be five main categories of techniques used more frequently, or a combination of them.

- The first is the network flow methods. In these methods a transportation problem is solved using a minimum cost flow or shortest path algorithm. The core area is divided into core bins, and density values are calculated for each bin based on the cell area in it. The cost of moving a cell to another bin is calculated according to the objective function. Generally, cells are moved from the densest bin to the less dense bins, following paths that result in the least degradation of placement quality.
- The second category, min-cut methods, separates the global placement into sets of equal number of paths and uses partitioning techniques to minimize their interconnections.
- The third category, linear placement methods, optimizes a linear arrangement of cells within a single row.
- Another basic category, random methods, such as one based on simulated annealing, randomly moves around locally to improve some objective function.
- At last, some greedy approaches are used for wirelength optimization in the final placement stage.

More specifically, final placement can be divided in two steps, legalization and detailed placement wirelength optimization.

4.2.1 Legalization

Given an illegal placement (constraints described in chapter 1), legalization is the process that eliminates all overlaps by perturbing the modules as little as possible.

Removing overlaps is the first challenge for a detailed algorithm. The common technique for removing overlaps is by stacking cells along rows or spreading out cells within all placement bins. Another approach is stacking cells row by row without inserting white space and without consideration of the imbalance. This approach was first met in Feng Shui algorithm.

In a different method that was met in Domino algorithm, an overlap-free compact placement is constructed by growing the placement like a crystal region by region from left to right.

Another approach grows the placement cell by cell from left to right and it performs a less restrictive version of packing cells to the left, introduced in Tetris algorithm. Sometimes, detailed placement is separated in two stages with a coarse legalizer such as Mongrel making substantial improvements in overlap removal before a fine-grain legalizer is finally used.

4.2.2 Local improvement

Minimizing wirelength or other objectives during detailed placement stage often is done by swapping target cells from different rows. Other cells in these rows might have to be shifted in order to remove the possible overlap or white space.

Some placement tools consider only any combination of cells' order within one single row. Domino improves the current wirelength driven placement solution iteratively.It also reduces overlaps by separating the core area into bins and solve the problem locally, each of which is formulated as a transportation problem solved by network flow methods. Then only within a row cell swaps are allowed, avoiding extensive shifting of cells. Another placement technique proposes using dynamic programming approaches to separate the cells within a row into two groups dynamically changed, and finally rearrange them with a near optimal way in the row. However this technique has extra complexity for multiple numbers of rows. Capo uses a branch-and bound end placer to find the best permutation of cells in a small region. It divides and stitches white space with cells if there is white space with the region, thereby distribution the cells in the region evenly.

Some placement use cells' swapping between different rows. For instance, during the Dragon2000's detailed placement swaps cells in different without any consideration of objective value's changes, and in most of the cases other cells have to be shifted to refine the placement, which sometimes results worst placement solutions. Another approach that was met in FengShui, swaps cells in different rows only when this swap results in no overlap with other cells in order to avoid shifting other cells. However these movements limit the effectiveness of inter-row cell swaps. Another related issue is that the calculation of accurate wirelength changes for shifted cells might be computational expensive. Timberwolf uses estimated wirelength changes of those shifted cells to reduce computation cost.

4.2.3 Simulated annealing

As introduced in previews paragraph Simulated annealing (SA) is a generic probabilistic method for the global optimization problem. It results a good approximation to the global optimum of an objective function in a large, discrete search space with many local minima. In many cases Simulated annealing may be more efficient than any other heuristic.

The main idea of this algorithm has its roots in annealing in metallurgy, a technique uses heating a material up to a critical temperature and controlled cooling it to increase the size of its crystals and change properties such as hardness and ductility. The heat treatment make the atoms unstuck from their electron shell with the minimum internal energy and wander randomly through states of higher energy. Then, with the controlling cooling, atoms try to return in low internat energy states and sometimes lower than the initial one.

Simulated annealing is one of the most well developed placement methods available. It is used in placement as an iterative improvement algorithm. Given an initial placement, a change is made by moving a component or by swapping locations in two components.

```
Algorithm Simulated-Annealing
Begin
      temp=Init-Temp;
      place=Init-Placement;
      while(temp>Final-Temp)do
            while(inner_loop_criterion = False)
                  new_place=Pertrub(place);
                  ΔC=Cost(new_place)-Cost(place);
                  if(ΔC<0) then
                        place=new_place;
                  else if(Random(0,1)>e^(ΔC/temp))then
                        place=new_place;
            temp=Schedule(temp);
End
```

Picture 13: Simulated Annealing algorithm.

All moves that result in a decrease in cost are obviously accepted. Moves that result in an increase of cost are accepted with a probability that decreases over the iterations. The analogy to the actual annealing process is heightened with the use of a parameter called temperature T, which parameter controls the probability of accepting moves with increased cost. The initial value of the temperature in the algorithm is very high which gradually degrades during the process so that the moves that increase cost have lower probability of being accepted. Finally, the temperature reduces to a very low value which causes only moves that reduce cost to be accepted.

By using this process simulated annealing can exceed local minimums and find the global minimum or near global minimum solutions. Parameters and functions used in

a simulated annealing algorithm determine the quality of the placement. These include the cooling schedule consisting of initial temperature, final temperature and the function for changing temperature. A good choice of parameters and functions can result in a good placement in a relatively short time.

The method was independently described by Scott Kirkpatrick, C. Daniel Gelatt and Mario P. Vecchi in 1983, and by Vlado Černý in 1985. The method is an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by M.N. Rosenbluth in a paper by N. Metropolis in 1953.

Simulated annealing is used mostly in detailed placement; nevertheless Sarrafzadeh and Wang use this method on a global placement after the placement area is partitioned into bins. Each move in the simulated annealing implementation of them is a swap of cells between bins. The size of the grid is determined from analysis of the circuit.

The method was also used in Dragon algorithm also by Sarrafzadeh and Wang at the stage of the detailed placement. The first stage of the placer in consists of a hierarchical approach. The placement area is recursively divided into a 2x2 bin structure. This recursion ends when there are only seven components in each bin. At each level of the hierarchical algorithm a bin swapping algorithm attempts to swap the contents between neighboring bins to improve the placement.

4.2.4 Greedy Approaches

Greedy approaches, usually, check every possible swap of cell or group of cells, in order to find the best position for them by respecting the local constraints set.

Detailed placement can also be met as final or absolute geometrical placement. The amount of detailed placement depends on the type of global placement algorithm used. In difference to the global placement stage, detailed placement does not attract much attention due to its relative low run-time complexity. However, a bad detailed placement may fail to preserve or improve the quality of a good global placement. In this chapter there will be reviewed several techniques that remove residual overlaps that remain after global placement and improve further the half the total wirelength locally, in the detailed placement stage.

4.3 Placement Algorithms

In this section very significant placement algorithms, which are usually the basis of optimized ones, are described. Their main goal is to reduce total wirelength.

4.3.1 Min-cut

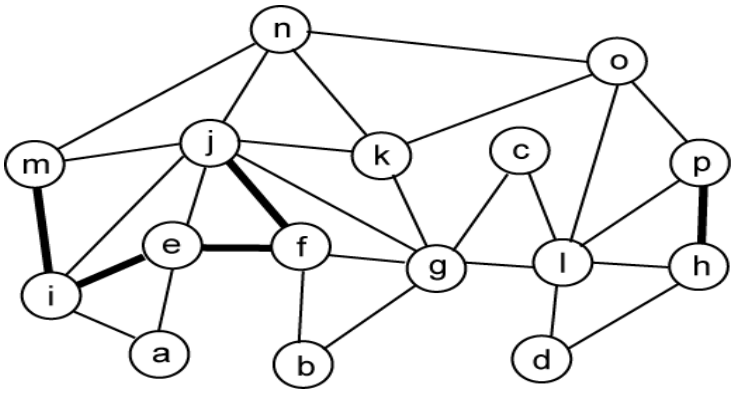
The min-cut algorithm developed by Breuer, in 1977, is based in partitioning to perform placement and it is a global placer. In this method the circuit given is repeatedly divided into submodules. Correspondingly the layout region is divided horizontally or vertically to accommodate the submodules. Partitioning is repeated until its partition is occupied by a single gate or a small submodule. In case of former, the placement is legal with no overlaps. In case of latter, legalization is performed in order to find unique location for the cells. The pseudocode is presented below

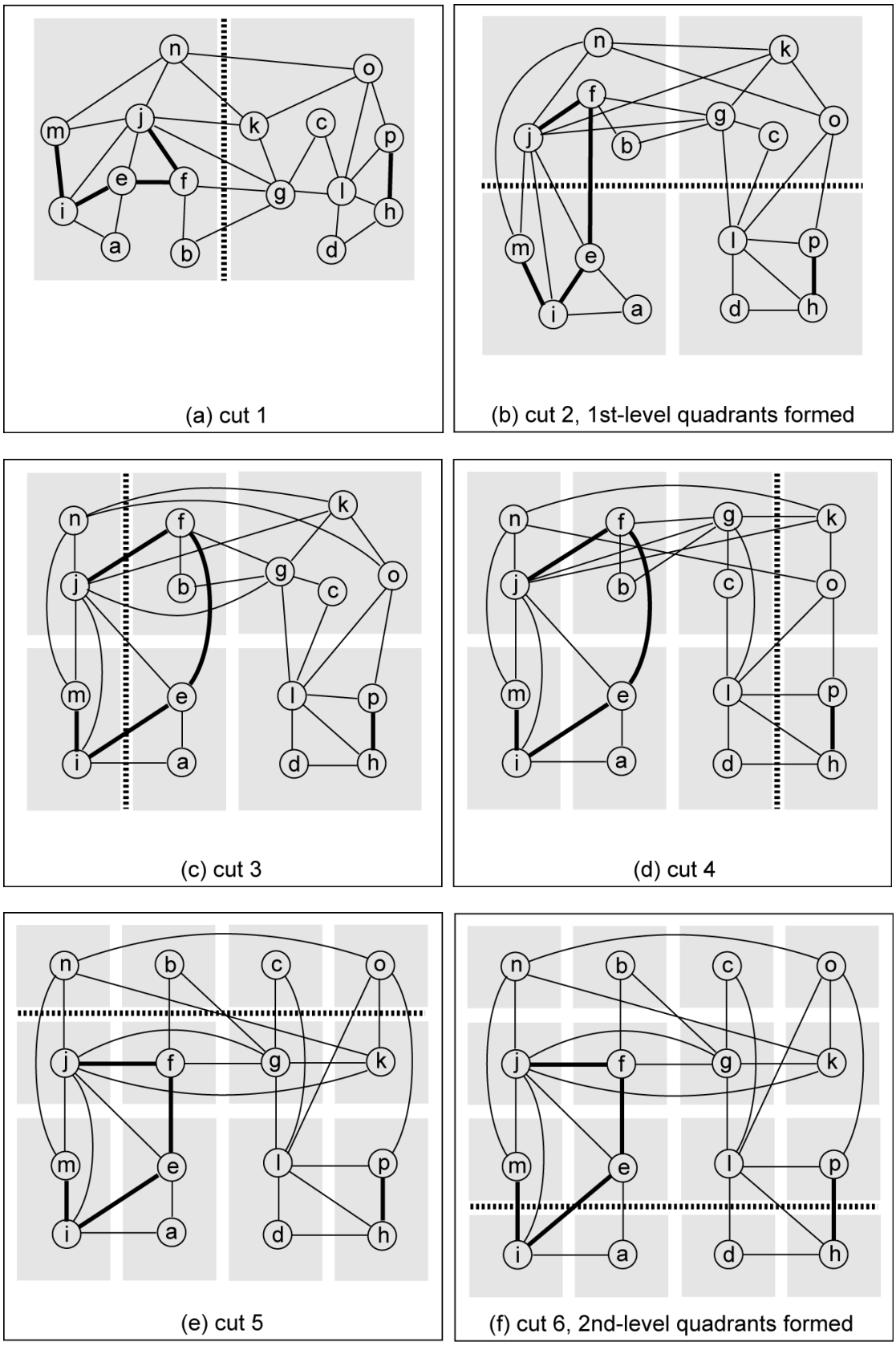
```
Variables: queue of placement bins
Initialize queue with top-level placement bin
1.While (queue not empty)
2.         Dequeue a bin
3.         If(bin small enough)
4.              Process bin with end-case placer
5.         Else
6.              Choose a cut-line for the bin (including
  direction)
7.              Build partitioning hypergraph from netlist and
  cells contained in the bin
8.              Partition the bin into smaller bins
9.              Enqueue each child bin
```

Picture 13: Min-Cut algorithm.

Breuer suggested that the cutsize should be minimized during the partitioning, which minimizes the overall wirelength, and presented several “cut orientation sequences” to decide how to partition the region. In the following pictures it is presented how the min-cut works with quadrature placement, where the placement region is divided into four partitions by a pair of vertical and horizontal cut.

$n_1 = \{e, f\}$
$n_2 = \{a, e, i\}$
$n_3 = \{b, f, g\}$
$n_4 = \{c, g, l\}$
$n_5 = \{d, l, h\}$
$n_6 = \{e, i, j\}$
$n_7 = \{f, j\}$
$n_8 = \{g, j, k\}$
$n_9 = \{l, o, p\}$
$n_{10} = \{h, p\}$
$n_{11} = \{i, m\}$
$n_{12} = \{j, m, n\}$
$n_{13} = \{k, n, o\}$





Picture 15: Min-Cut example [17]

4.3.2 Gordian

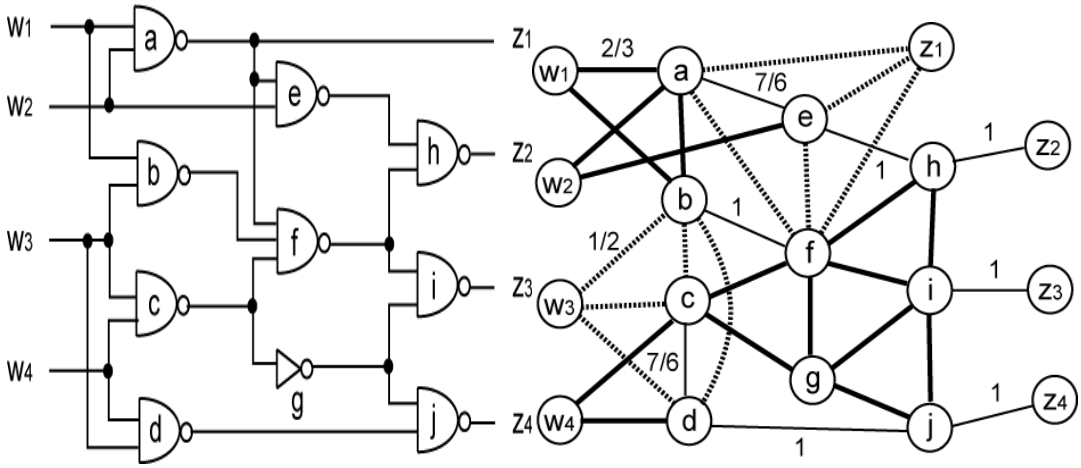
Gordian developed by Kleinhans et al., in 1991, is one of the most successful algorithms that adopt quadratic methods into circuit placement problem and it is a global placer. In Gordian, the problem is formulated as sequence of quadratic programming, derived from the connectivity information of the circuit. Gordian is an iterative algorithm, so in every iteration, a new set of constraints are imposed in order to limit the freedom of movement of the cells, causing the placement solution to reduce the amount of overlap of the cells. During Gordian, top-down partitioning is performed. At this step the "center of gravity" constraint is satisfied by the cells grouped into the same partition, i.e. the center of the gates' partition are drawn by location's partition center. Iterations ate repeated until the size of the partitions is small enough. After Gordian terminates there may be overlap among the cells. Therefore a post-process such as Domino, which will be analyzed at a later stage, is performed in order to legalize the placement. The pseudocode is presented below

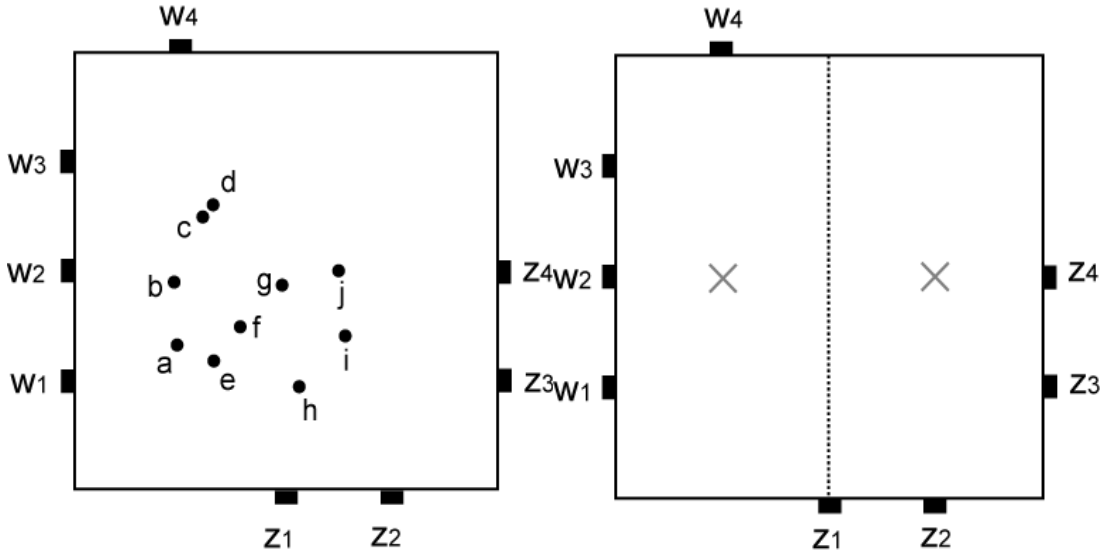
```
Gordian Procedure:
l := 1
global_optimize( l );
while ( there exists |Ml|>K )
        for each r
        partition( r, r', r'' );
        l ++;
        setup_constraints( l );
        global_optimize( l );
        re-partition( l );
final_placement( l );
end_procedure;
```

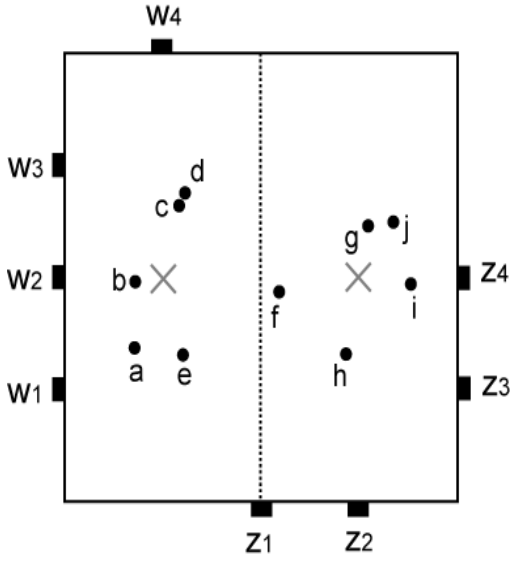
Picture 16: Gordian algorithm.

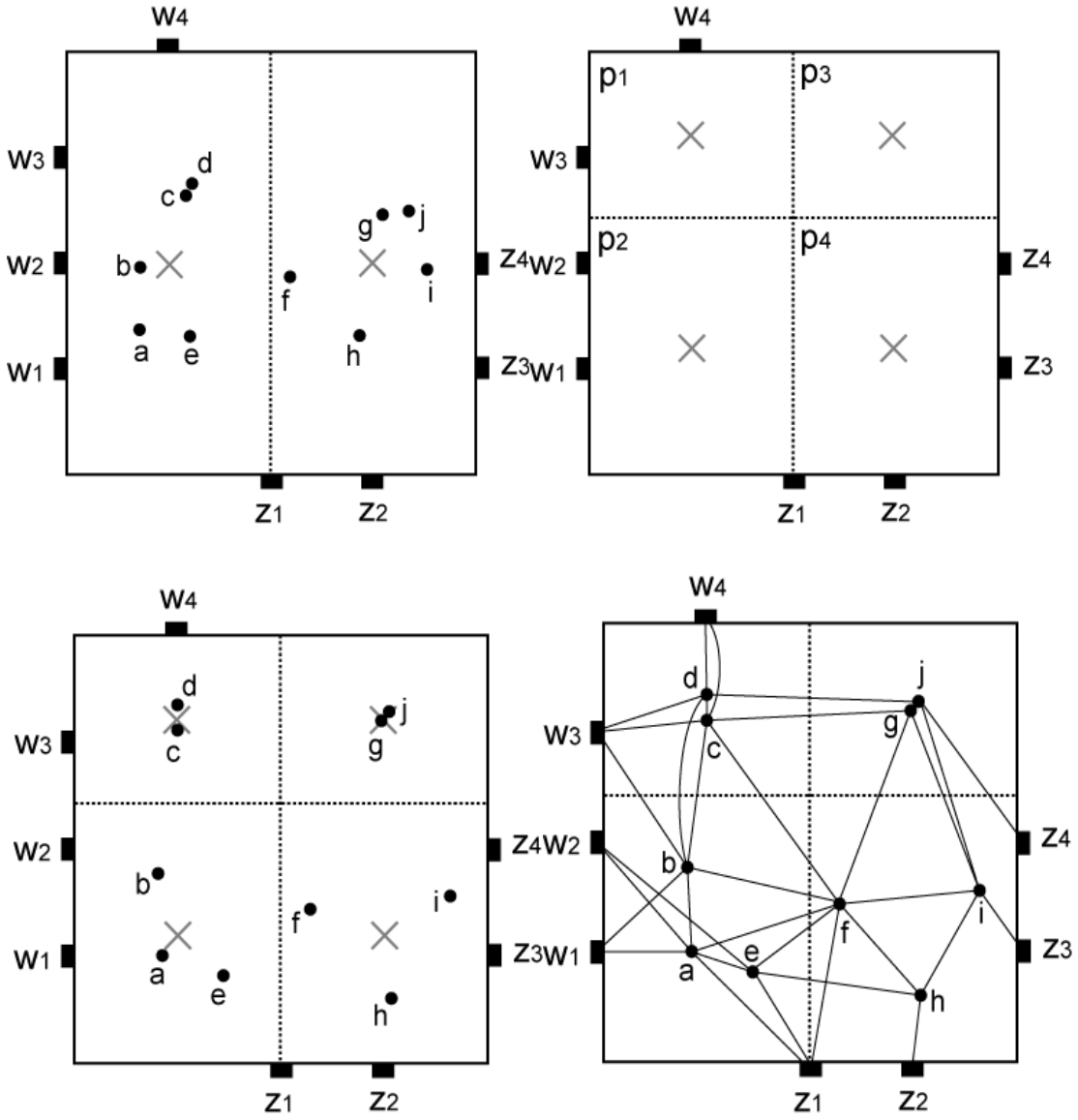
The goal of Gordian is reduce the total wirelength. In order to do so, it uses a quadratic objective function, to minimize the squared wirelength among the cells. Gordian odds from other QP/partitioning based placers, as it uses QP as a global optimizer, instead of local regions.

A run of Gordian placement is displayed in the pictures below.









Picture 17: Gordian example[17]

4.3.3 TimberWolf

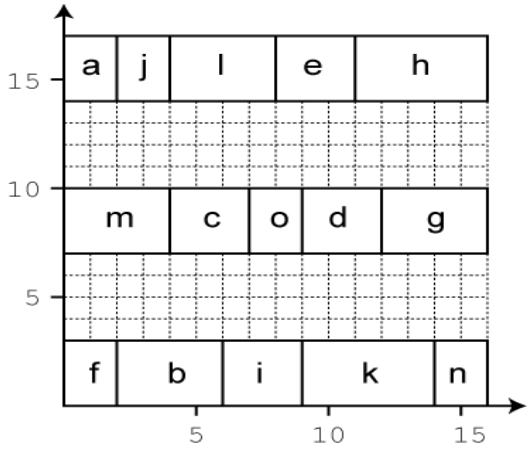
TimberWolf algorithm developed by Sechen and Sangiovanni-Vincentelli in 1985 and is a detailed placer based on Simulated Annealing. A hierarchical placement approach is used, where the netlist is clustered twice in a recursive function. The second-level clusters are first placed during the high annealing temperature region. Then these clusters are decomposed to reveal first-level clusters, which are refined during mid annealing temperature region. Finally, first-level clusters are decomposed to the original gate level netlist, which is placed during the low annealing temperature.

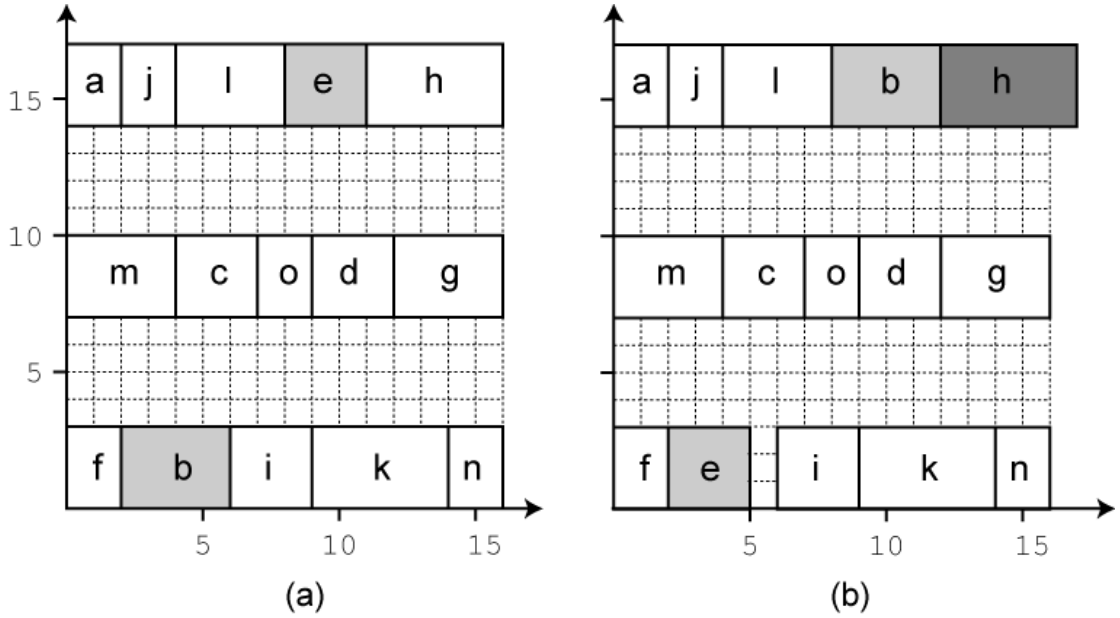
Since the overlap removal is a time-consuming process, TimberWolf allows cell overlap and tries to minimize it by utilizing a penalty function. In case the overlap is

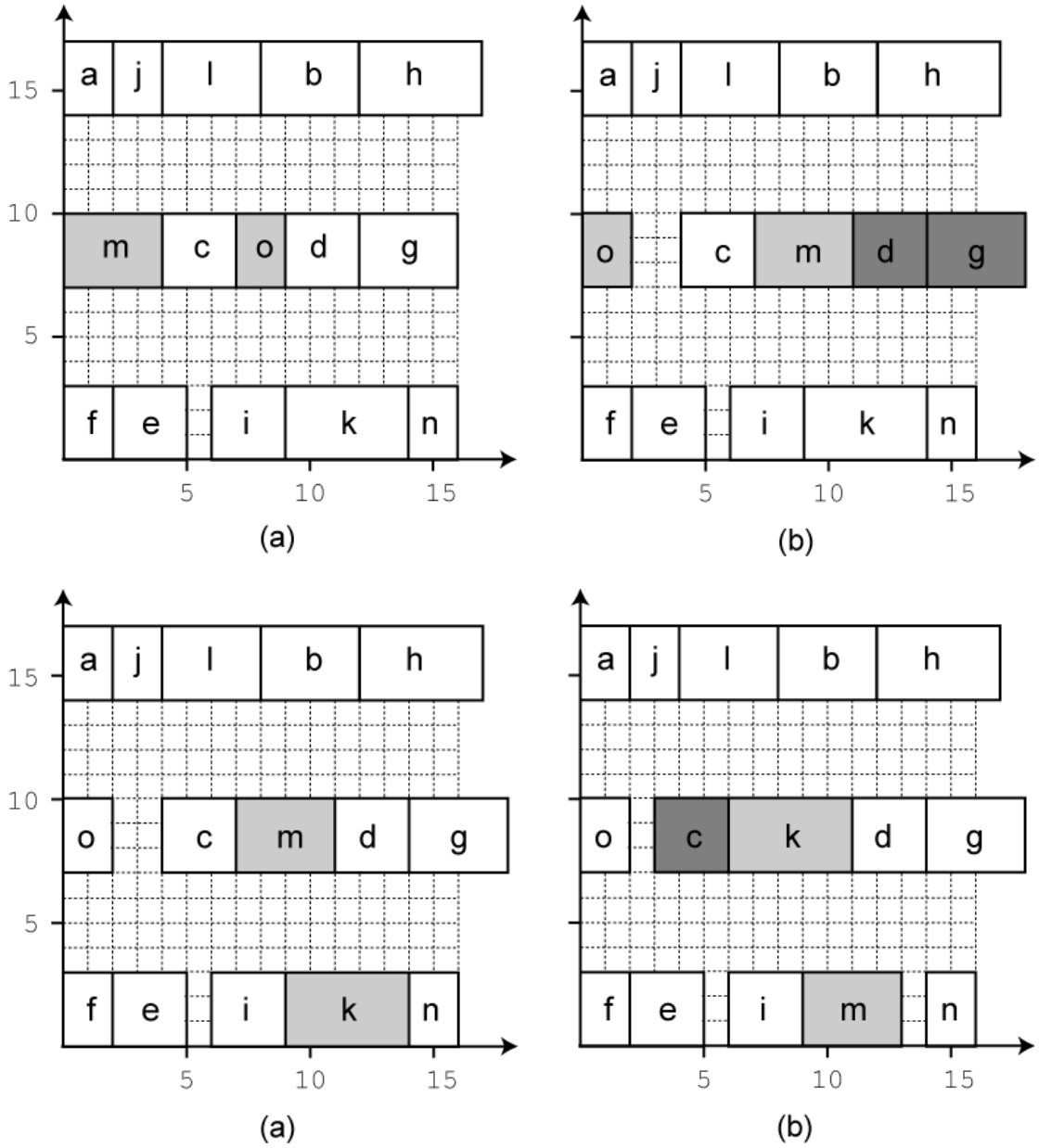
not removed, cells are shifted at the end of the process to obtain a legal placement, therefore the solution quality degrades.

Because of the degradation, the authors of TimberWolf 7.0, Sun and Sechen in 1995, proposed a way to maintain overlap-free placement during annealing, while performing cell shifting efficiently. Steps are presented below.

$n_1 = \{a, e, g\}$
$n_2 = \{f, o\}$
$n_3 = \{b, c, k, n\}$
$n_4 = \{d, h, i\}$
$n_5 = \{j, l, m\}$
$n_6 = \{d, k, j\}$
$n_7 = \{c, e, f, h, n\}$
$n_8 = \{d, l\}$
$n_9 = \{b, g, i, m\}$
$n_{10} = \{a, k, o\}$







Picture 18: Timberwolf example[17]

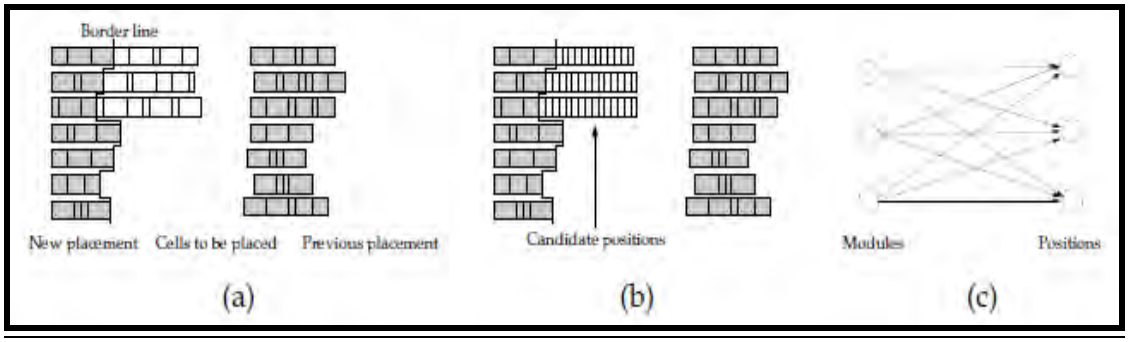
4.3.4 Domino

Domino is a detail placer pioneered by Konrad Doll, Frank Johannes and Kurt Antreich in 1994.Domino uses as input a placed netlist, with or without overlap, such as the output of Gordian. Then an iterative process produces a sequence of intermediate placements. In each iteration, an improved placement is generated without overlap, as the placer divides the layout into regions and solves a set of local subproblems. For each region, cells are assigned into new positions by formulating a transportation problem, according to a cost function approximating wirelength. Domino pseudocode is described below.

```
Algorithm Domino
Begin
Initial placement;
while(improvement)
      for (all_regions)
            get_cells;
            provide_locations;
            solve_transportation_problem;
            move_cells;
      end
end
adjust_row_lengths;
swap_cells;
End
```

Picture 19: Domino algorithm.

The basic idea is that through iterations Domino can avoid local minimum that may degrade the final placement solution.



Picture 20: Domino example

4.3.5 FastDP

Fast DP is a detail placer pioneered by Min Pan, Natarajan Viswanathan and Chris Chu in 2005. FastDP uses a legalized placement as input and consists of 4 techniques. The first one is Single-segment clustering to obtain a good starting solution for the main steps of the algorithm. Then there is a loop that performs Global swap, Vertical swap and Local reordering until there is no significant wirelength improvement. Finally Single-segment clustering is reapplied to get better position for the cells within the segments, without changing their order.

Global swap is used to find the "optimal region", that is the region where the wirelength is optimal. After the optimal region is found, penalty on overlap is introduced, as there may be the possibility that the gate cannot be moved to this area, as the initial placement is legal.

To increase the possibility of a good swap, Vertical swap is a technique similar to Global swap but not such greedy. With this algorithm local vertical errors are fixed, but for local horizontal ones Global swap is too expensive. Therefore a Local reordering is introduced where all possible left-right ordering of the cells is tried and returns the best wirelength. The pseudocode of FastDP is presented below

```
Algorithm FastDP
Begin

Perform Single-Segment Clustering
Repeat
        Perform Global Swap
        Perform Vertical Swap
        Perform Local Re-ordering
Until no significant improvement of wirelength
Repeat
        Perform Single-Segment Clustering
Until no significant improvement of wirelength
```

Picture 14: Fast DP algorithm.

4.3.6 Proud

The eigenvector analysis approach is far too slow on large circuits. Instead a faster quadratic clique formulation can be used. The main idea is based on reformulating the problem as a linear resistive network problem which can be optimized by solving a system of linear equations. This is done with Successive Over Relaxation (SOR), which method is a predecessor of optimizing by Conjugate Gradients. As mentioned earlier such a placement is likely overlapping and therefore a partitioning method is used. A vertical cut line is moved from left to right until the sum of the area of the modules on the left of the cut line is roughly half. This divides the set of modules in two groups. The modules are confined to two regions each corresponding to half the core area and the method now optimizes each group while considering the modules of the opposite group as fixed. Their coordinates are simply projected onto the region boundary. Proud is related to the graph partitioners but use the analytic placement to guide the cut instead of the min-cut objective. The method proceeds recursively on each region alternating between vertical and horizontal cut lines until only one module remains. A heuristic four-way partitioner is also proposed and test runs show that the four-way partitioner gives improved netlength but often uses twice the time or more.

4.3.7 The Vygen's Method

Another partitioning method was published by Vygen. First a quadratic placement is calculated by minimizing the clique netlength. Then the circuit is divided into four parts and the position of the modules from the quadratic solution is used to determine into which of the four parts each module is to be placed.

The splitting algorithm gives close to minimal total movement of the modules and runs in linear time. The process is repeated but for the subsequent quadratic optimizations the nets are split so that no net cross a region. If a net crosses a boundary of a region an artificial module is placed on that boundary. This leads to a new formulation

for each net on each region containing only articial and ordinary modules of that region.

Before partitioning at each level a re-partitioning step is also conducted on each $2x2$sub-grid in the current grid. The modules of the $2x2$ subgrid are thrown together again and a quadratic problem for the $2x2$sub-grid is solved. After partitioning this leads to a new placement of the modules in the sub-grid which is accepted if it is better than the old one. Extra care is taken to avoid situations where all modules in the same region end in the same spot. This is done by introducing a constraint that makes the center of gravity of the modules be at the center of each region. Vygen's algorithm can also place large macro modules in combination with standardcells.

This is done by including them in the partioning step. When regions become too small to contain the macros a branch-and-bound algorithm is used to place all macros within the region while minimizing total movement. The algorithm is similar to that of Onodera et al., but here branching occurs on the two most overlapping modules. The nodes of the branch-and-bound algorithm are linear programs which are duals of minimum cost flow problems and can be solved in $O(nlogn(m + nlogn))^5$ time.

The final phase of the algorithm considers moving modules between the final regions. Regions are characterized as having too many modules if they cannot contain the modules. Also some regions may have surplus space. The movement of modules is modeled as a minimum-cost-flow problem between adjacent regions. A knapsack problem determines which modules to move. Vygen’s method can handle both standard-cells and fixed-cell-layout.

4.4. Performance driven Placement

The delay at chip level, which depends on the interconnection network, has a major role in determining the performance of the chip. As technology advances, now it is at deep sub-micron designs, the size of the chip decreases and the wires become a major factor for high-performance chips. The placement algorithms for high performance chips have to generate placements that allow routers to route nets within the timing requirements. These problems are called performance driven placement.

These algorithms can be classified into two categories:

- Net-based approach
- Path-based approach

Net based approach, tries to route the nets to meet the performance constraints on individual nets and the algorithm has to decide the performance requirement for each net. On the other hand, the path-based approach, takes account of the critical paths in the circuit and tries to place the blocks in such a manner, that the path length is within its performance constraint.

4.5 Trends

In Very Deep Sub-Micron designs, placement problem is considered much more than simply achieving the routability of the design and minimizing the chip die area. Several other critical issues such as timing, voltage drop, even power distribution are increasing the complexity of the placement problem exponentially. Since placement takes place in the early phase of the physical design, lot of attention it is paid.

Algorithms to estimate the wirelength of the circuit are becoming part of placement algorithms, because the accurate estimation help to fix problems in placement step and not in routing step. Estimation of the wirelength helps to understand the routability of the design.

The estimation of the wirelength is discussed in:

1. Early placement to obtain better Wire Load Models (WLM) for synthesis, as it is a parameter for the delay estimation during synthesis.
2. Placement for Crosstalk avoidance
3. Placement for minimizing clock-skew
4. Placement for even power distribution

5. Wirelength driven placement

In the following chapter are presented the main features and important properties of the code that was implemented, in order to address some of the major problems seen in modern approaches of the placement problem. After that, our placement algorithm will be presented.

5.1 Preparation

In the following sections we will make a detailed description of the files that are used as input at the platform developed, and the output of it as well as industrial tools that were used.
The algorithm was implemented using the programming language C.

5.1.1 Input files

The input of the algorithm we developed is according to the standards being used by industrial placement tools and described in chapter 2:

- **Verilog netlist**
- **Technology library:** Nangate 45nm Open Cell Library, which is an open cell library
- **I/O pin placement:** The positions of I/O pins are being placed randomly by the algorithm or a file that described their position can be given as input.

5.1.2. EDA Tools

The tools being used by this thesis is Synopsys's Design Compiler that synthesizes the RTL netlist and produces the Verilog netlist according to a usual script used for synthesis.
For the placement of the circuit were used an industrial tool Cadence's Encounter and an academic tool, Capo, which is among the best academic tools for placement. These tools were used for the comparison of the results.

5.1.3 Output

The output of our algorithm is a .txt file that describes the final position of the cells of the IC. The user has the capability to see the results through graphic representation by using the GLUT package.

5.2 Levelization – A wirelength driven placer

In this section we will present the approach we have followed to implement a

wirelength-driven placer, a tool responsible for the proper placement of the cells of a circuit in such way that the total wirelength required for the interconnection of components to be as minimum as possible.
The algorithm receives as input a netlist and the technological library used for the synthesis. In the first step of the algorithm these files are parsed and the useful information they contain is being stored in appropriate data structures, in such way that they can be fast tracked.
Thereafter, the size of the total surface, on which the placement algorithm is going to place the standard cells, is calculated. The input and output pins are considered as entities without actual dimensions which are placed in the region at the periphery of our space.
The total surface location can either be defined by the user, or the default event to be calculated using the following formula:

$$placement\_area = (1+(1-utilization)) \times total\_cell\_area$$

where utilization is at 90% in our tests, because we want to push the algorithm and the tools to their limits.

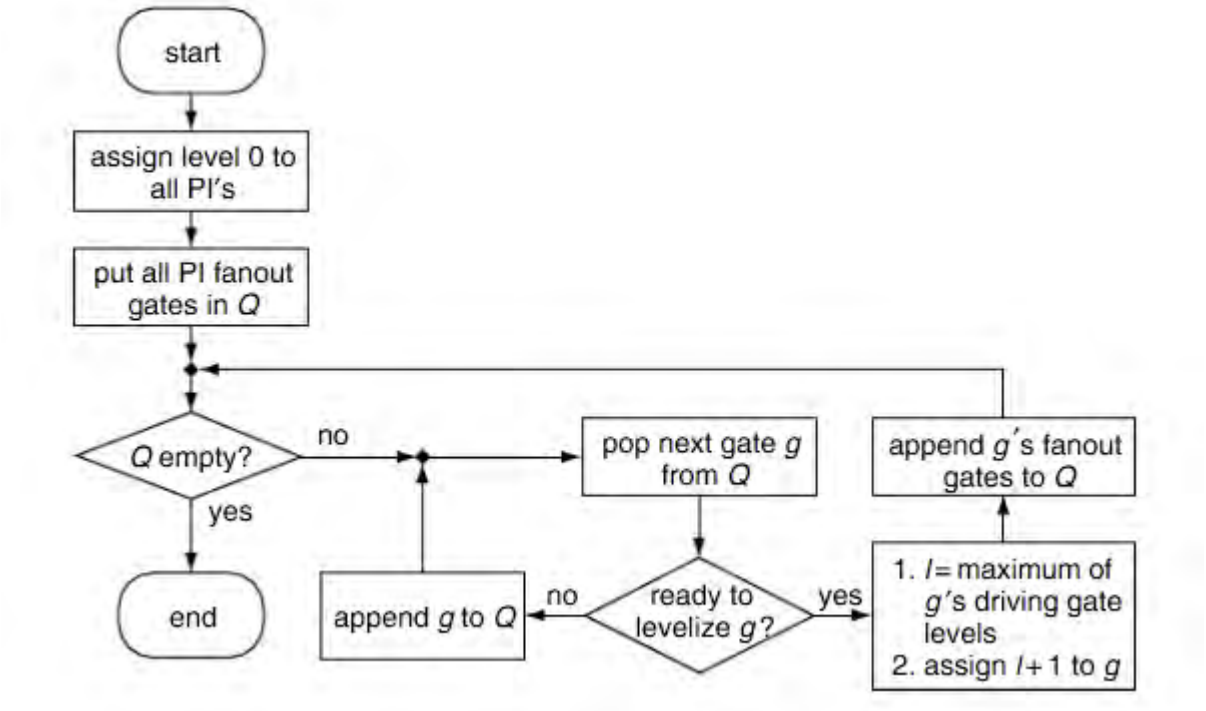
The calculation of the sides of the rectangle to be used is based on the placement_area. The gates to be placed are standard cells, i.e. the height is fixed and they differ only in their width. Therefore, the height of the rectangle must be a multiple of the height of the standard cells.
At the next step the initialization of the coordinates of input and output pins is taking place, according to the methods described above.
Then the basic code starts. As first step is the levelization process which is described later in this chapter. After finding the level of each gate, the gates are being placed in the area according to their level. Here it must be mentioned that the selection according to the level can be done with three different methods, from the first to last, from the last to the first or simultaneously. These three different methods provide different results, due to the complexity of the circuit. Therefore, it was decided that the algorithm should run all the methods and as result it provides the best. The algorithm's steps will be presented at a later section.
Thereafter a legalization step is taking place. The legalization being performed is similar to ABACUS legalizer. At the final step, a detail placer similar to FastDP, is being performed.
All these steps will be analyzed in the sections following bellow. It must be mentioned that for the comparison of the results is done by comparing the total wirelength of the circuit, which is calculated by the bounding box metric described in chapter 3.

5.2.1 Levelization technique

The levelization technique is used when logic gates must be evaluated in an order such that a gate will not be evaluated until all its driving gates have been evaluated. Fr circuit N, the evaluation order:
G1→G2→G3→G4

satisfies this requirement.

The logic levelization algorithm can be utilized to produce the desired gate evaluation order. At the beginning of the algorithm, all the primary inputs are assigned level 0, and all the primary inputs fanout gates are appended to the first-in/first-out queue Q that stores the gates to be processed. While Q is non-empty, the first gate g in Q is popped out. If all the driving gates of g are levelized and the maximum level is l, g is assigned level l+1, and all the fanout gates of g are appended to Q; otherwise, g is put back in Q to be processed later. The levelization process repeats until Q is empty. Note that for gates assigned the same level, their order of evaluation does not matter. This levelization process is also referred as "rank ordering".



Picture 22: Levelization algorithm

The levelization process for circuit N is shown step by step below. At the begging, primary inputs are assigned level 0, and their fanout gates G1 and G2 are appended to

Q. In step 1, G2 is not ready and put back to Q because G1 is not levelized yet. In step 2, G1 is assigned level 1because it is driven by level 0 primary inputs only. At the end of the process, the following orders are produced:

G1→G2→G3→G4

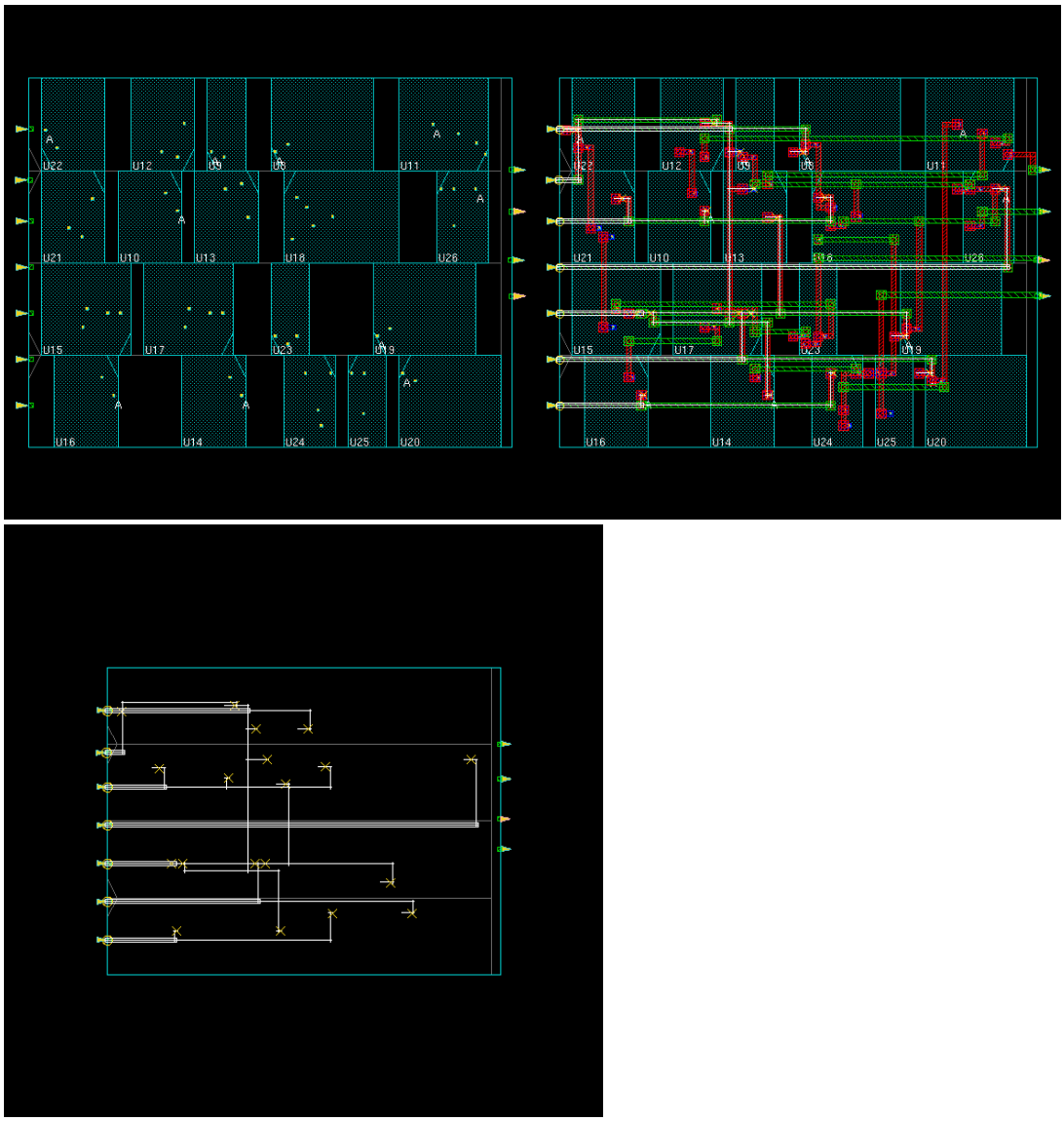
G1→G3→G2→G4

Step	A	B	C	G_1	G_2	G_3	G_4	Q
0	0	0	0					$<G_2, G_1>$
1	0	0	0					$<G_1, G_2>$
2	0	0	0	1				$<G_2, G_3>$
3	0	0	0	1	2			$<G_3, G_4>$
4	0	0	0	1	2	2		$<G_4>$
5	0	0	0	1	2	2	3	< >

Picture 23: Levelization example [42]

5.2.2 Level placer

Lever placer is a constructive wirelength driven placer. The main idea of this algorithm is to start placing the standard cells according to their levelization value, increasingly. The explanation for that is that when primary inputs and outputs are placed, level 0 gates that are connected only with inputs are placed first, nearby the primary inputs, then level 1 gates that are highly connected with primary inputs and level 0 gates are placed closed to both etc. The effect of this algorithm, is that highly connected objects are placed closed, and with a constructive way it finds a near optimal wirelength solution. Further if primary inputs are placed-fixed on one side of the core area and the primary outputs are placed-fixed on the opposite, a vertical line could separate the core area into two parts, where ideally every module of the right part could have greated level value of every cell of the left part.



Picture 24: The first picture shows a cell placement. The second picture shows a cell placement and metal routing. The third picture shows the wires from inputs to cells. The commercial tool for this circuit gives great wirelength to these wires.

Above is shown a fixed pins wirelength driven placement, from the Cadence SocEncounter tool. The most of net that connect the primary inputs with gates grow in the left half of the core area. This fact leads to the main idea of the levelization algorithm that brings the highly connected cells near the primary inputs, and the cells that have greater level value on the right of them.

Level placer algorithm's steps (step 4 changes according to the level selection mentioned earlier):

1. Parsing the netlist, and computing the core area to 90% utilization of the total standard cell area calculated from the using standard cell area, and ratio 1.

2. Seperating the netlist's modules in levels, with the levelization technique described above.
3. Pins are considered as fixed before the algorithm starts. The optimistic way for this algorithm leads to placing the primary inputs on one or two sides of the core area, and the outputs on the opposite side/sides. The pesimistic is placing the primary input and outputs randomly, peripherally of the core area.
4. Start placing modules with the smallest level number first, finding a non-legal position with the smallest bounding box wirelength. These positions are calculated such as the x of the cell to be the expected value of its inputs' x, and the y of the cell to be the legal expected value y of its inputs' y – a legal standard row position-.
5. After all modules are placed, a legalization algorithm is performed.

The fourth step described above, is for the FIRST LEVEL placer. The second one called LAST LEVEL placer, makes exactly the same function but it starts for the last level to the first. Finally, there is the MIXED LEVEL placer that places simultaneously 2 levels one from the smallest and one from the higher level, until the middle level is reached.

5.2.3 Legalizer

The legalizer used is similar to ABACUS. The cells are placed by the LEVEL placer at a legal core row, but they may have overlap. In order to fix the overlap a legalizer is taking place. The legalizer starts by sorting the cells of a row and then tries to find a legal position in the row. If there is no space in the row the cell is sent one row up, with the same coordinate. This is repeated for each row until there is no overlap. The pseudocode is:

```
1. For each row:
2. Sort the cells by their x-coordinate
3. For each cell
        1. Place from the left to the right making the minimum movement, as possible
        2. If overlap, solve by moving both cells
        3. If no space in row, move the right cell one row up, with the same x-
coordinate.
4. Until no overlap in row
5.Repeat
```

Picture 15: Legalization algorithm.

By using this algorithm, the final placement, is legalized and our circuit is ready to advance at the detailed placer for further improvement of the total wirelength.

5.2.4 Detail placer

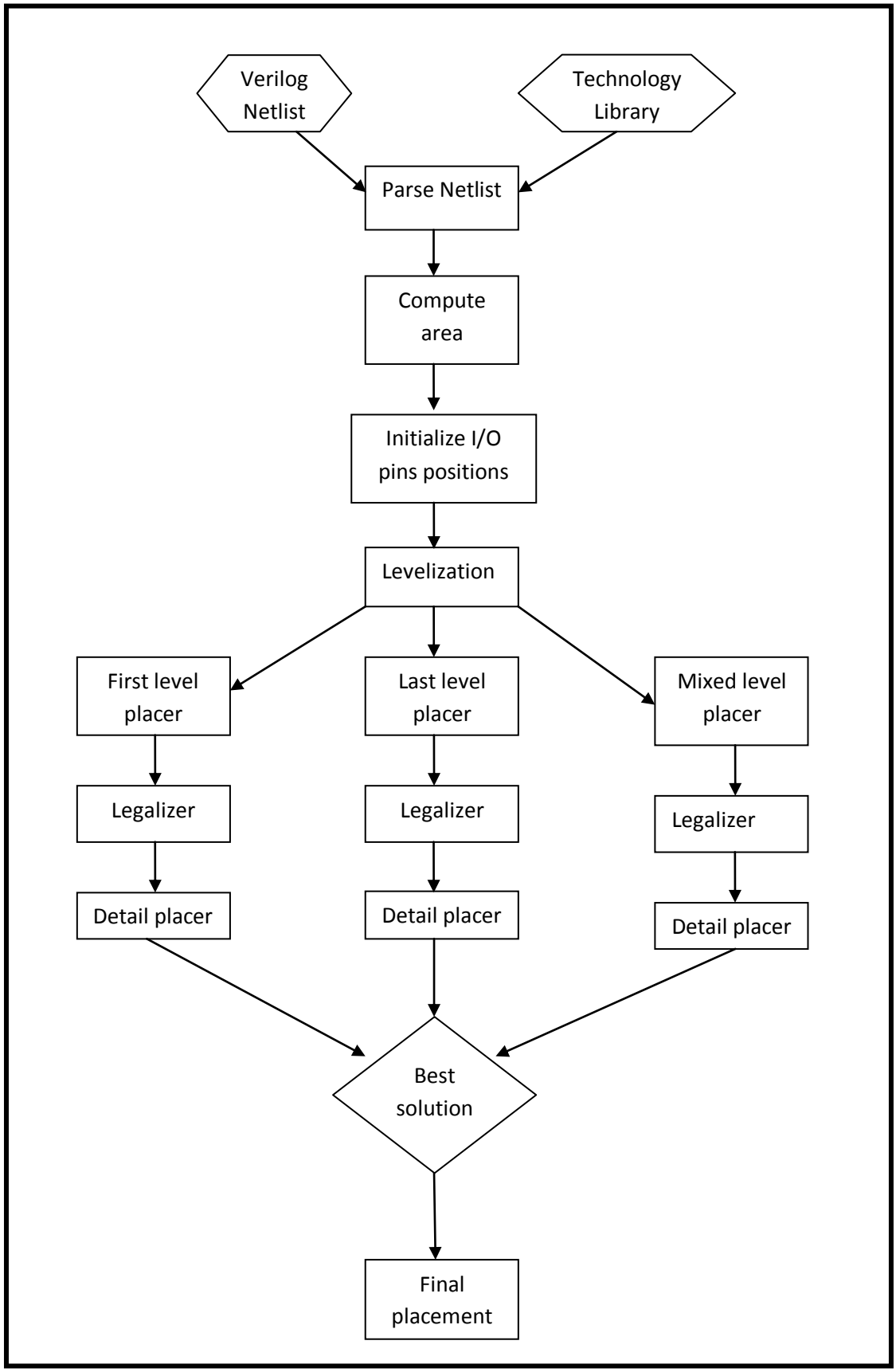
As last step of the algorithm we developed is a detail placer similar to FastDP. The algorithm has as input the legalized placement produced by the legalizer above. The first thing that does is a global swap, i.e. the cells are being replaced in the area according to the best position to be placed, as described in FastDP. Then a vertical swap is taking place, i.e. the cell is replaced at its best y-coordinate. After that, a local swap, in the row, is being performed in order to place the cell at its best position in the row. Finally, the legalizer described above takes place in order to make sure that there is no overlap. The pseudocode is:

```
1. Perform Global swap, until no further wirelength
improvement.
2. Perform Vertical swap, until no further wirelength
improvement.
3. Perform Local swap, until no further improvement.
4. Legalize
```

Picture 16: Detailed placement algorithm.

5.2.3. Synopsys

At this section we present the algorithm as whole.



Picture 17: Our wirelength placement solution.

6. Benchmark Circuits

The determination of the performance of an EDA tool is one of the major issues to be faced by a designer. One widely distributed way to control the final result is the selection of appropriate benchmark circuits, which will be given as input to the software. In this thesis are used the ISCAS '89 and ITC '99 benchmark circuits. In the remaining part of the chapter basic characteristics of these circuit designs will be presented, as well as, statistical data concerning the number and type of modules that the circuits are consisted of.

ISCAS '89: The circuit descriptions of ISCAS '89 circuits are provided in both structural and behavioral in form. All these high-level design models have been proven very useful as research tools in many areas of digital circuit design and more particularly in test generation, the procedure of timing analysis and technology mapping. In the official documentation of the benchmarks is given a set of additional information, beyond the descriptions in language material. We will limit ourselves to present at table 1 the number of primary inputs / outputs and cells contained in each of the circuits in this group.

Circuit	FF's	PI's	PO's
S298	14	3	6
S349	15	9	11
S382	21	3	6
S386	6	7	7
S400	21	3	6
S420	16	19	2
S444	21	3	6
S510	6	19	7
S526	21	3	6
S641	19	35	24
S820	5	18	19
S832	5	18	19
S838	32	35	2
S953	29	16	23
S1196	18	14	14
S1238	18	14	14
S1423	74	17	5

S1488	6	8	19
S1494	6	8	19
S5378	179	35	49
S9234	211	19	22
S35932	1728	35	320

Table 1: ISCAS circuits

ITC ’99: The need for benchmark circuits which will not be quite small or quite simply, led to the creation of a suite that includes a set of digital designs, which were collected from companies whose research coinciding with the subject area of digital integrated circuit design. The key features of the ITC '99 benchmark circuits are:

• Circuits are completely synthesizable using Synopsys Design Compiler.
• Does not contain compiler specific instructions.
• All packages required by the descriptions of circuits in HDL language is either arithmetic packages either IEEE standard logic packages.
• The global reset signal is always available.

The circuit descriptions, contained, vary from simple "monolithic" circuits (1 entity, 1 process) to multiple circuits and processes. More specifically, one of the circuits contained in this group are three times larger than the largest ISCAS '89 (37 inputs, 69.917 gates, 3.320 flip-flops).

Below is a table showing the number of gates, primary inputs / outputs and flip-flops contained in each of the circuits.

Circuit	Gates	PI’s	PO’s	FF’s
B01	49	2	2	5
B02	28	1	1	4
B03	160	4	4	30
B04	737	8	11	66
B05	998	1	36	34
B06	56	2	6	9
B07	441	1	8	49
B08	183	9	4	21
B09	170	1	1	28
B10	206	11	6	17
B11	770	7	6	31
B12	1.076	5	6	121

B13	362	10	10	53
B14	10.098	32	54	245
B15	8.992	36	70	449
B17	32.326	37	97	1.415
B18	114.621	36	23	3.320
B19	231.320	21	30	6.642
B20	20.226	32	22	490
B21	20.571	32	22	490
B22	29.951	32	22	735

Table 2: ITC circuits

7. Experimental Results

ISCAS Circuits	MIXED	FIRST	LAST	BEST
S27	82,662941	78,941811	81,034996	*FIRST*
S298	484,75824	489,660095	501,94043	*MIXED*
S344	1366,25281	1417,90027	1366,77551	*MIXED*
S349	1174,27332	1178,24194	1038,33289	*LAST*
S382	810,147583	767,368469	849,434204	*FIRST*
S386	928,044739	907,46637	877,853455	*LAST*
S400	872,505493	877,608887	873,04248	*MIXED*
S420	1322,49951	1366,70874	1299,15405	*LAST*
S444	1111,41101	1126,54688	1026,45447	*LAST*
S510	2247,78809	2226,97461	2234,17334	*FIRST*
S526	1082,03125	1079,66345	1071,71106	*LAST*
S641	2263,92554	2253,85767	2225,94605	*LAST*
S713	2234,65845	2228,89575	2064,60107	*LAST*
S820	2133,98071	2135,8667	2089,27026	*LAST*
S832	2512,25293	2572,36523	2480,63281	*LAST*
S838	3916,86548	3986,73535	3984,48096	*MIXED*
S953	4031,30322	4125,55078	3904,25781	*LAST*
S1196	5364,98438	5439,57031	5412,51563	*LAST*
S1238	5581,46826	5512,43994	5449,56494	*LAST*
S1423	6107,44092	6062,67773	6910,09229	*FIRST*
S1488	5965,02637	5958,03223	5908,26514	*LAST*
S1494	7431,26807	7426,18652	7337,23633	*LAST*

Table 3: Levelization algorithm results

ISCAS Circuits	Capo	Industrial placer	LEVEL PLACER
S27	69,946663	74,341652	78,941811
S298	380,065338	402,505768	484,75824
S344	877,462524	802,012634	1366,25281
S349	882,345093	720,410034	1038,33289
S382	576,385498	589,55542	767,368469
S386	561,108459	554,263306	877,853455

S400	590,660645	582,493652	872,505493
S420	820,19696	849,882202	1299,15405
S444	765,115967	745,48584	1026,45447
S510	1146,781494	1093,166382	2226,97461
S526	732,682495	668,801941	1071,71106
S641	1395,674927	1255,704224	2225,94605
S713	1393,410156	1240,200928	2064,60107
S820	1341,569336	1212,894043	2089,27026
S832	1479,520142	1436,810181	2480,63281
S838	1821,942749	1513,420532	3916,86548
S953	2657,027344	2476,232666	3904,25781
S1196	3191,276367	2729,357178	5412,51563
S1238	3470,307861	2965,37915	5449,56494
S1423	4178,119141	3505,828613	6062,67773
S1488	3627,42627	3255,343994	5908,26514
S1494	4140,110352	3648,336914	7337,23633

Table 4: Results comparison

Our results as are seen in the table above are not promising at first glance. Nevertheless let us give a clearer overview of the results as was understood of our experience.

Capo was introduced in 1999, and the commercial tool in the decade of ’90. They have years of experience, work and improvement to achieve these results. However, comparing Capo editions we have seen great improvement in its results, up to 40%. We also can see from the table that Capo is better than the commercial tool. It can be explained by considering that commercial tool tries to find a multi-metal, free of violations result. It tries to leave space between nets and pins to avoid DRC and other violations.

There are many reasons to explain our results. First of all the other tools use a combination of many global placement algorithms, to achieve a good result for placement, in difference with our placement tool that uses only one global placement algorithm, which is first introduced by our team.

On the other hand, our global placement algorithm is constructive. That means that it runs in O(n), in difference with iterative algorithms that that run in O(mn) where m is the number of iterations. Additionally, partitioning and clustering also have O(mn) complexity. The disadvantage of constructive algorithms is that they stack in local minima, which can get passed with iterative algorithms like simulated annealing, and lead to a better result.

Another thing of consideration is that how pins are placed. Commercial tool firstly places the cells, and at the end places the pins in coordinates that give the minimum wirelength, and the same stands for Capo. Our tool uses fixed pins taken their coordinates from the commercial tool. This way our pin placement is not optimal. Moreover, levelization technique is stand in the fact that cells are placed from left to right for level 0 to n, so it would be for efficient if all input pins were left and all output pin were right. But we used commercial tools coordinates for pins to count the worst case of our placement tool, which exists by non-optimized pins. We wanted our results to be realistic about modern circuits which have many primary inputs and have them spread in every side of the circuit, and modern circuits that are not rectangle shaped.

In chapter “Future work” we suggest some parts that could be inserted in our tool, in order to improve further wirelength, and give more competitive results for wirelength.

8. Power-aware driven placement

In the context of IC power optimization there are two different ways to reduce the power consumption of a chip; static power and dynamic power. Static power is the power dissipation for DC supply and is due to leakage current. Dynamic power is the power dissipation from the change of a transistor's state.

Static power does not depend on the cell location; therefore placement cannot affect static power. Multiple voltages can be used in order to reduce the static power consumption, by changing voltage-island assignments. On the other hand, dynamic power depends on interconnect lengths, which are determined by placement. To support higher clock frequencies, modern designs are heavily pipelined. In order to support design scaling, placers must optimize

- Hundreds of signal nets, each consuming a small amount of power.
- Clock networks with significant power consumption.

Static power reduction techniques are trading positive timing slack for power. When voltage-islands are present, cells can be moved closer to voltage sources in rows and in regions. When in rows are in interleaving rows of high and low V_{dd} rows. When in regions, cells are powered by the closest voltage island. There are and approaches that include cell hierarchy, clustering and locality.

Dynamic power consumption can be accomplished by reducing net activity, or optimizing register locations, or optimizing the clock tree. These classes are not mutually exclusive.

Later, in this thesis are referred only dynamic power techniques, as our idea is about reducing total wirelenght.

8.1. Power aware placement review

In this section are presented three algorithms based on weights on signal nets.

8.1.1."Temperature aware global placement"

This algorithm, pioneered by Obermeier and Johannes at TUM, approaches the temperature awareness by the power dissipation. More specifically they try to minimize the total power dissipation and arrange the cells in a way such that the resulting temperature profile is flatten.

So they expressed the power dissipation of a logic gate as

$$P= 0.5*C_{load}*V_{dd}^{2}*f*E$$

on average. V_{dd} is the voltage of the power source, f is the clock frequency and E is the switching rate. $C_{load} = C_{pin} + C_{net}$, and $C_{net} = l_{net} * c$, where l_{net} is the length of the Steiner tree and c is the capacity per unit length.

Then a temperature analysis is performed, where power dissipation is used in the cost function.

8.1.2. "Tabu search"

Another approach for power driven placement is by using Tabu Search, thus by incorporating fuzzy logic in the design of aggregating function.

At this algorithm, cost functions for wirelength and power consumption and timing are evaluated and co-exist in order to be optimized simultaneously. This is the reason of using fuzzy logic, to integrate these multiple, possibly conflicting objectives into a scalar cost function. The rule created is:

IF a solution has
SMALL wire length AND
LOW power consumption AND
SHORT delay
THEN it is an GOOD solution.

"Tabu Search is an elegant heuristic that proceeds by making iterative perturbations while preventing cycling to certain number of recently visited points in search space."[45]

```
Algorithm Tabu_Search
Ω    :   Set of feasible solutions
S    :   Current solution
S*   :   Best solution
Cost:    Objective function
N(S):    Neighborhood of S ∈ Ω
V*   :   Sample of neighborhood solutions
T    :   Tabu list
AL   :   Aspirartion level
Begin
Start with random initial solution S ∈ Ω
Initialize tabu list and aspiration level
For fixed number of iterations Do
    Generate neighbor solutions V* ⊂ N(S) by swapping
        two randomly chosen cells
    Find best S* ∈ V*
    If move S to S* is not in T Then
        Accept move and update best solution
        Store index of a swapped cell in T
    Else
        If Cost(S*) < AL Then
            Accept move and update best solution
            Store index of a swapped cell in T
        End If
    End If
End For
End.
```

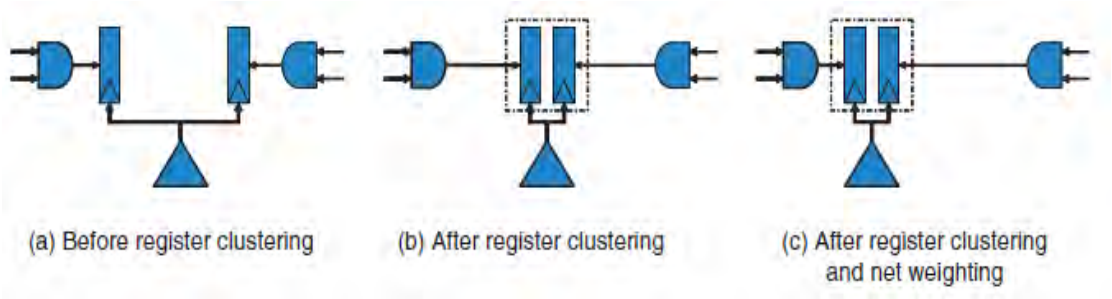
Picture 18: Tabu search algorithm

8.1.3. Power aware placement

One final approach for power placement is one that simultaneously performs activity-based register clustering and activity-based net weighting. The first parameter reduces clock power by placing registers in the same leaf cluster of the clock trees close together and the second parameter reduces net switching power by assigning a combination of activity and timing weights to the nets with higher switching rates or critical timing.

The net switching power dissipation can be modeled as $P=kCV^2a$, where k is a constant, C is the total capacitance, V is the power supply and a is the activity factor. The power aware placement aims at reducing the net power by reducing the product Ca.

So the power-aware algorithm combines the register clustering and the net activity-based net weighting to simultaneously reduce the clock and signal net switching power. The picture 29 below represents the effect of the algorithm.



Picture 199: Power aware placement steps.

8.2 Our approach

Our concept about the power aware driven placement is to try minimize

$$P= V^2 *f *\Sigma_k (C_k*af_k)$$

where V is the supply voltage, f is the frequency, $\Sigma_k C_k$ is the summary of the capacitance for each node, called the total switch capacitance and af is the activity factor for each node. Additionally, $C_k = C_{gk} +C_{wk}$, where C_g is the capacitance of the gate and C_w is the capacitance of the wire of the interconnection of the gate. Therefore, the minimization can be done if we use af*wirelength as cost function. This can be achieved by using known placement algorithms such as GORDIAN and use as weight the cost function mentioned above. Unfortunately, we can't present any results as our algorithm is not finished yet and the way of evaluation hasn't been decided yet.

9. Future Work

Our work can be used as a platform for developing new placement tools, academic or industrial. Nevertheless, there are a lot of things that can be implemented in order to improve this algorithm. All the work to be done is summed up bellow:

- Implementation of additional methods for net modeling representation.
- Use multi-net models while processing circuit.
- Revise the existing placement algorithm and implement an iterative part, in order to avoid the local minimum that the constructive algorithm may have found.
- Implementation of additional methods of legalization.
- Implementation of an algorithm, that finds the global minimum wirelength. There is some work done about it, but we didn't present any results, as there are a lot of things that must be reviewed. This algorithm can be used to check the quality of our final placement results.
- Implementation of timing and power driven placement.
- Creation of a graphical interface.
- Implementation of a router for the design.
- Parallelization of algorithms.

10. Bibliography

[1] Myung-Chul Kim, Dong-Jin Lee and Igor L. Markov. SimPL: An Effective Placement Algorithm. , Department of EECS, University of Michigan. IEEE, 2010.

[2] Charles J. Alpert , Dinesh P.Mehta and Sachin S. Sapatnekar. Handbook of Algorithms for Physical Design Automation. Chapter 21 Timing driven placement. Auerbach Publications 2008.

[3]Jens Egeblad. Placement Techniques for VLSI Layout Using Sequence-Pair Legalization. Master of Science Thesis,Department of Computer Science University of Copenhagen. 2003.

[4] Taraneh Taghavi, Xiaojian Yang and Bo-Kyung choi, Dragon2005: Large-Scale Mixed-size Placement Tool. CS Dept. UCLA, Synplicity Inc, Magma Design Aytomation Inc. 2005.

[5] Ameya R. Agnihotri , Satoshi Ono and Patrick H. Madden. Recursive Bisection Placement: Feng Shui 5.0 Implementation Details. SUNY Binghamton CSD and university of Kitakyushu. 2005.

[6] Natarajan Viswanathan and Chris Chong-Nuen Chu. FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement, and a Hybrid Net Model. IEEE transactions on computer-aided design of integrated circuits and systems, vol 24, no 5, May 2005.

[7] Jun Wang and Edmund Y. Lam. Standard Cell Layout With Regular Contact Placement. IEEE transactions on semiconductor manufacturing, vol 17, no 3, August 2004.

[8] Jens Vygen and Bernhard Korte. Flow-based Partitioning and Fast Global Placement in Chip Design Dissertation. Rheinischen Friedrich-Wilhelms university, Bonn, Juli 2010.

[9] Evriklis Kounalakis, Christos P. Sotiriou. SCPlace: A Statistical Slack-Assignment Based Constructive Placer.FORTH-ICS, Heraklion, Crete, Greece. 2011.

[10] Ameya Agnihotri, Mehmet Can, YILDIZ Ateen Khatkhate, Ajita Mathur, Satoshi Ono, Patrick H. Madden. Fractional cut: Imprved recursive bisection placement. SYNY Binghamton Computer Science Department. 2003.

[11] Naveed Sherwani. Algorithms for VLSI Physical Design Automation –Third edition. Chapters 5-7. eBook ISBN 0-306-47509-X. 1999.

[12] Zhong Xiu, Rutenbar. VLSI Component Placement by Grid Warping. Chapter 2. Carnegie Mellon University Pittsburgh, Pennsylvania. 2003.

[13] Konrad Doll, Frank M. Johannes, and Kurt J. Antreich. Iterative Placement Improvement by Network Flow Methods. IEEE transations on computer-aided design of integrated circuits and systems, vol 13, no 10. October 1994.

[14] Min Pan, Natarajan Viswanathan and Chris Chu. An Efficient and Effective Detailed Placement Algorithm. Department of Electrical and Computer Engineering, Iowa State University. 2005.

[15] Ulrich Brenner, Anna Pauli, and Jens Vygen. Almost Optimum Placement Legalization by Minimum Cost Flow and Dynamic Programming. Research Institute for Discrete Mathematics, University of Bonn. 2004.

[16] Charles J. Alpert, Gi-Joon Nam and Paul G. Villarrubia. Free Space Management for Cut-Based Placement. IBM Corporation. 2002.

[17] Sung Kyu Lim. Practical problems in VLSI physical design automation. Chapters 1-4. e-ISBN 978-1-4020-6627-6. 2008.

[18] Zhe-Wei Jiang, Hsin-Chen Chen, Tung-Chieh Chen, and Yao-Wen Chang. Challenges and Solutions in Modern VLSI Placement. Institute of Electronics Engineering, National Taiwan University. 2007.

[19] J. Bhasker and Rakesh Chadha. Static Timing Analysis for Nanometer Designs. A Practical Approach. Chapter 2-5. e-ISBN 978-0-387-93820-2. 2009.

[20] Andrew B. Kahng, Chul-Hong Park, Puneet Sharma, and Qinke Wang. Lens aberration aware timingdriven placement. In *Proc. Design, Automation and Test in Eurpoe*, pages 890–895, 3001 Leuven,

Belgium, Belgium, 2006. European Design and Automation Association.

[21] Ameya R Agnihotri. Combinational Optimization techniques for VLSI Placement. Chapters 1-3. 2007.

[22] Chen Li. Placement of VLSI Circuits.Thesis, Purdue University. 2006.

[23] Brent Goplen. Advanced placement techniues for future VLSI circuits. Thesis, Unversity of Minnesota. 2006.

[24] Upavan Gupta . Utilitarian Approaches for Multi-metric Optimization in VLSI Circuit Design. 2009.

[25] Nadine Azemard, Lars Svensson. Integrated Circuit ns ystem Desing. Power and timing Modeling, Optimization and simulation. ISBN 978-3-642-17751-4. 2007.

[26] Yao-Wen Chang, Zhe-Wei Jiang, and Tung-Chieh Chen. Essential Issues in Analytical Placement Algorithms. 2007.

[27] Hidetoshi Onoderat, Yo Taniguchi, and Keikichi Tamaru. Branch-and-Bound Placement for Building Block Layout. 1991.

[28] U.Brenner J.Vygen. Faster Optimal Single-Row Placement with Fixed Ordering. 2000.

[29] Peter Spindler, Ulf Schlichtmann and Frank M. Johannes. Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement. Insitute for Electronic Design Automation Munich. 2008.

[30] Yu-Min Lee, Tsung-You Wu, and Po-Yi Chiang. A Hierarchical Bin-Based Legalizerfor Standard-Cell Designs with Minimal Disturbance. Department of Electrical Engineering National ChiaoTungUniversity. ASPDAC, January 2010.

[31] Ulrich Brenner. Theory and Practice of VLSI Placement. Phd thesis.2006.

[32] Tony Chan, Jason Cong, Kenton Sze. Multilevel generalized force-directed method for circuit placement. UCLA. 2005.

[33] Chen Li, Min Xie, Cheng-Kok Koh,J. Cong, P.H. Madden. Computer-Aided Design of Integrated Circuits and Systems. 2007.

[34] Jason Cong, Michail Romesis, Min Xie. Optimality, scalability and stability study of partitioning and placement algorithms. University of California at Los Angeles, Los Angeles, CA. 2003.

[35] Natarajan Viswanathan, Charles Alpert, Cliff Sze, Zhuo Li, Yaoguang Wei. The DAC 2012 routability-driven placement contest and benchmark suite. IBM Corporation, Austin, TX. 2012.

[36] Tsung-Yi Ho, Sheng-Hung Liu. Fast Legalization for Standard Cell Placement with Simultaneous Wirelength and Displacement Minimization. 2012.

[37] Igor L. Markov, Jin Hu and Myung-Chul Kim. Progress and Challenges in VLSI Placement Research. University of Michigan. 2012.

[38] Olivier Coudert. Gate Sizing for Constrained Delay/Power/Area Optimization. 1997.

[39] Charles J. Alpert, Anirudh Devgan, Stephen T. Quay, Buffer Insertion for Noise and Delay Optimization. IEEE. 1999.

[40] www.wikipedia.com

[41] www.asic-world.com

[42] Laung-Terng Wang, Cheng-Wen Wu, Xiaoqing Wen. VLSI test principles and architectures. 2006

[43] Yongseok Cheon, Pei-Hsin Ho, Andrew B. Kahng, Sherief Reda, Qinke Wang. Power-Aware Placement. DAC.2005

[44] Bernd Obermeier, Frank M. Johannes. Temperature-Aware Global Placement. ASP-DAC. 2004

[45] Sadiq M. Sait, Mahmood R. Minhas, Junaid A. Khan. Performance and Low Power Driven VLSI Standard Cell Placement using Tabu Search, CEC 2002