Department of Electrical and Computer Engineering

University of Thessaly, School of Engineering

Department of Electrical and Computer Engineering

Diploma Dissertation

Title: *"Graphical Representation of Standard-Cell Placement, a placement visualization tool"*

Student's Name: *George Stergioulas*

Supervisor: Prof.: *Georgios Stamoulis*
Assistant: M.Sc.: *Ioannis Arvanitakis*

Date: *October 2013*

Preface and acknowledgements

The following dissertation introduces a tool for the visual representation of the standard cells position within a chip area. It analyzes the placement problem in theory and the software design and functionality, acting also as a user's manual for the aforementioned tool.

This dissertation was made possible thanks to the idea and support of prof. Georgios Stamoulis and the invaluable assistance of MSc graduate Ioannis Arvanitakis. I was searching for a subject relevant to graphics, and the VEDA Laboratory of the Department of Computer and Communications Engineering at the University of Thessaly happened to work on placement algorithms development.

One of the design tools developed at the VEDA Laboratory needed a graphics interface so I started developing graphics for hardware design, leading to this dissertation. The finished result is a stand-alone GUI that can parse and draw the cell positions within a chip, after it has been provided with the appropriate data.

Finally I would like to express my gratitude to everyone that helped me reach this point, from fellow students and friends to professors. A separate thanks is also appropriate for Maria, whose support and guidance made all this possible.

Table of Contents

3. A story about graphics, CAD and EDA.

One of the most notable advances in computers in the past years must have been the way we perceive the information. From the command line interfaces of the early 80's, to the full-blown 3D games and interfaces.

Computer graphics by definition are in fact those generated from computers, but also the representation of image data using a computer system, with the assistance of specifically designed hardware and software.

This technology has led to the revolution of animation, movie production and of course the video game industry. Computers began as bulky monochrome boxes and forty years later they now fit in the palm of our hands, producing extreme computational power and impressive life-like graphics in real time. This is all relevant to the need for better and more sophisticated hardware design tools. Processors with billions of transistors keep reducing in size. They are becoming more and more efficient, as well as cheaper.
The future lies in the advances of embedded systems and mobile computing rather than big power-hungry desktops and the immersive technologies for seamless interaction with the virtual environments and augmented reality applications. This is what makes efficient and effective chip design techniques so important. More computational power is needed from smaller systems.

Some milestones in the development of graphics technologies and the way we interact with computers are shown below.

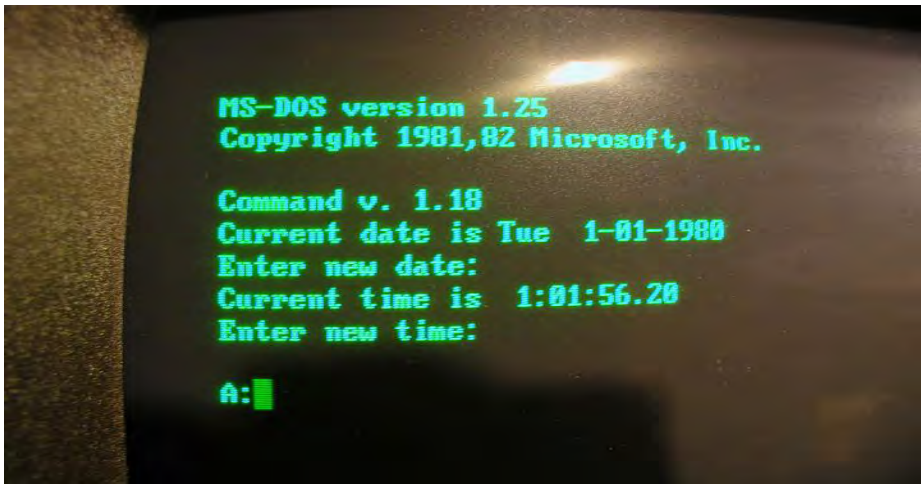
Picture 3.1

The original "Pong" game, 1972. The first mainstream video game in the world, by Atari.

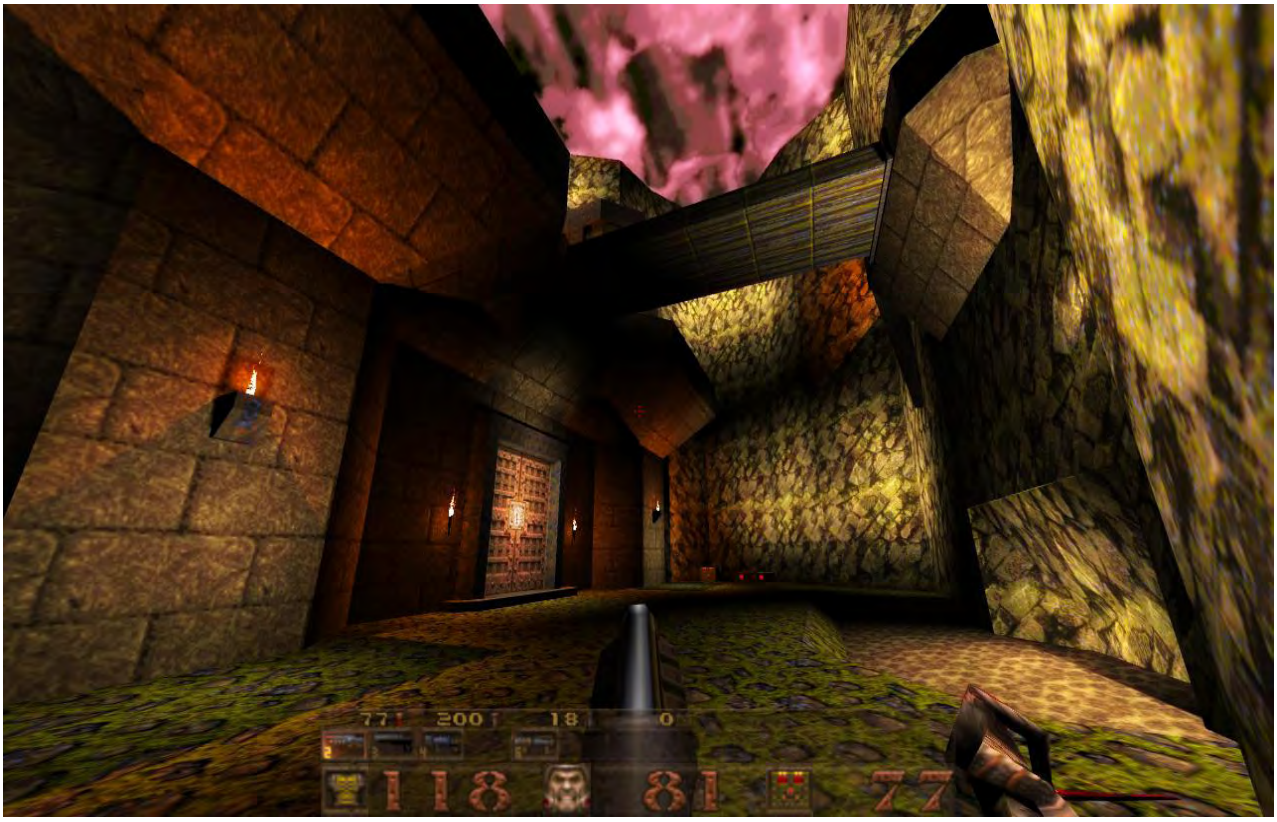
Picture 3.2

10 years after “Pong”, this command line interface on a monochrome screen is still the only way to interface with a computer

Picture 3.3



Picture 3.4



The year is 1996, and the world is presented with the first fully three dimensional game using graphics acceleration for the home computer. Quake is about to revolutionize the game industry and create a market for the graphics hardware industry as well.

The images above show the revolution that graphics brought to our world, and how fast and exponentially the whole technology evolved. From the 90's it only takes ten years for things to really take-off. Graphics hardware becomes impressively more powerful with every new generation affecting the world in multiple aspects. From the unmanned military drones to the entertainment industry, graphics technologies open up new possibilities.

Picture 3.5

In-game unedited screenshot showcasing how far graphics have come, from the “Project Cars” driving simulator.

Currently in early development stage it will be available within the current year for home PC’s.

Picture 3.6

The “Ironman” movies franchise, relies heavily on computer effects to create the immersive atmosphere. It also shows a plausible idea of how humans could interact with computers in the coming years.

This introduction leads us to the Graphical User Interface. In computing, graphical user interface (also known as “GUI”) is a type of human-computer interface that allows interaction with devices through graphical icons, windows, menus and visual indicators that can be manipulated by a mouse, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard. GUIs can also be manipulated by a keyboard although to a more limited extend. A window in a GUI is a part of the screen that can display information independently from the rest of the screen. Windows can open simultaneously and independently so that the user can organize the information within the space available on the screen. This space is usually called the desktop.

The actions in GUI are usually performed through direct manipulation of the graphical elements. Besides in computers, GUIs can be found in hand-held devices such as mobile phones, portable media players, gaming devices, household appliances, office, and industry equipment.

Picture 3.7

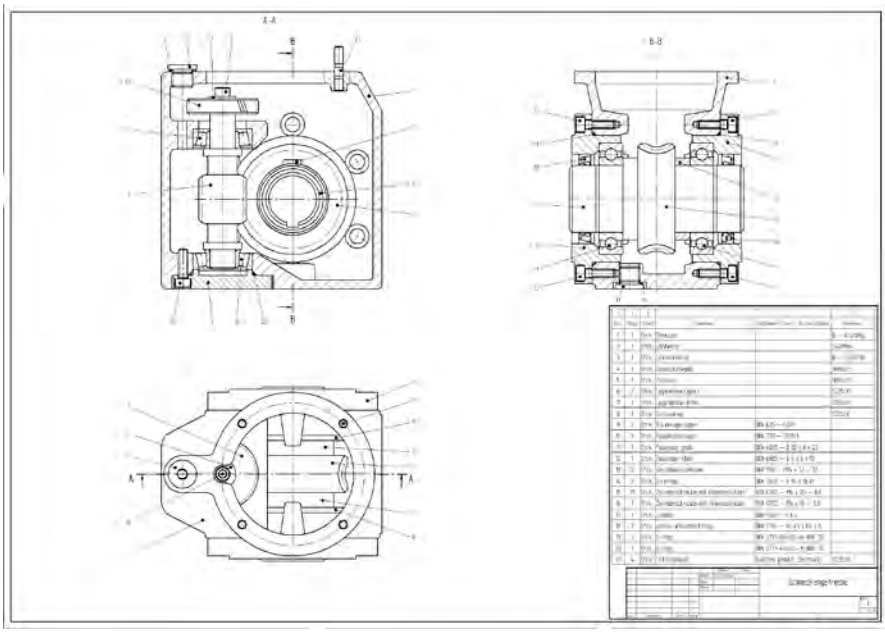
Command-line interface vs the latest Android touch interface

All of the above lead us closer to what this dissertation is about. GUI made the idea of Computer Aided Design or CAD possible. The definition of CAD according to wikipedia is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design. CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic

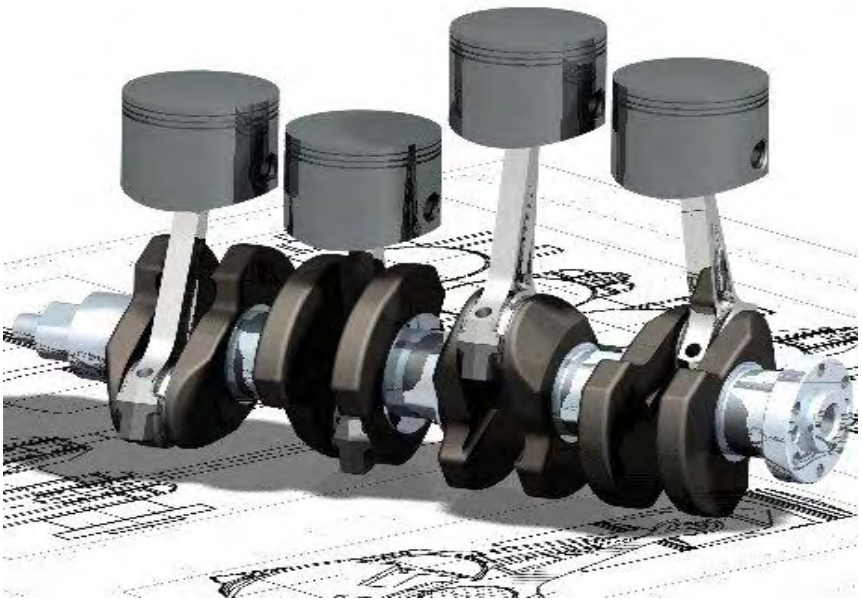
files for print, machining, or other manufacturing operations. Computer-aided design is used in many fields. Its use in electronic design is known as Electronic Design Automation, or EDA.

More about the history of CAD and EDA is to be found in the following chapter.

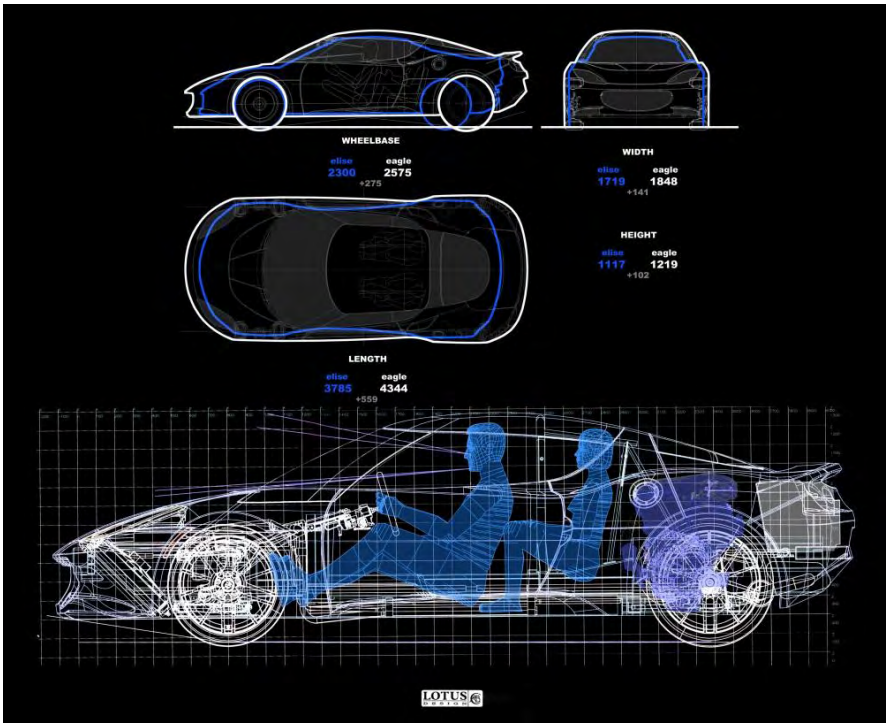
Picture 3.8



Picture 3.9



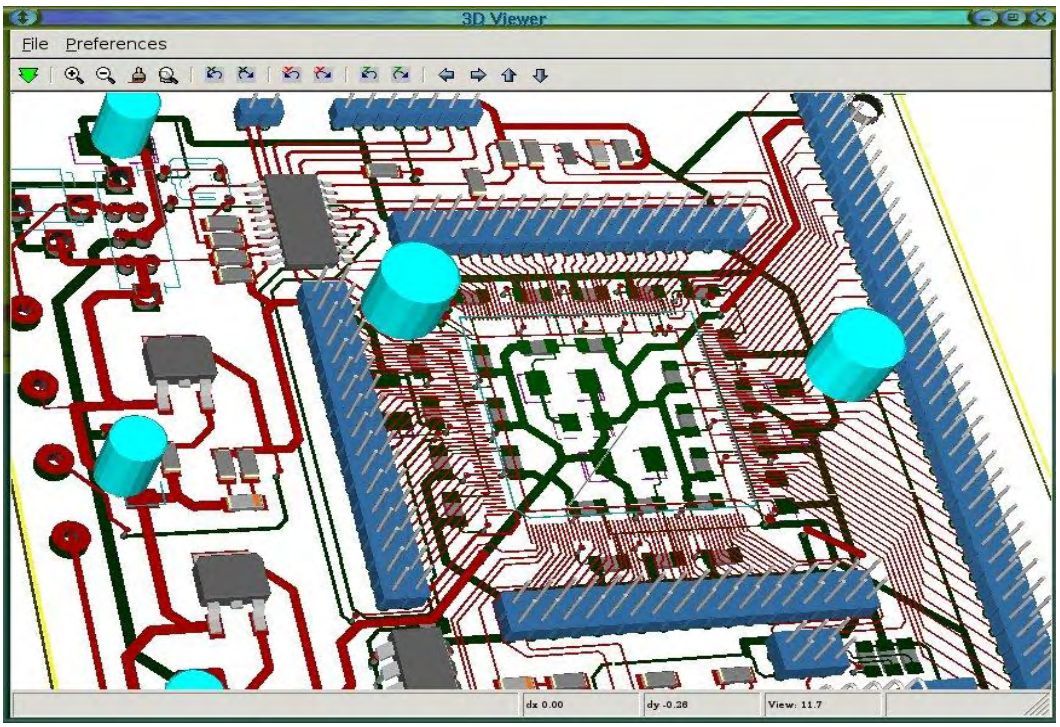
Picture 3.10



Picture 3.11

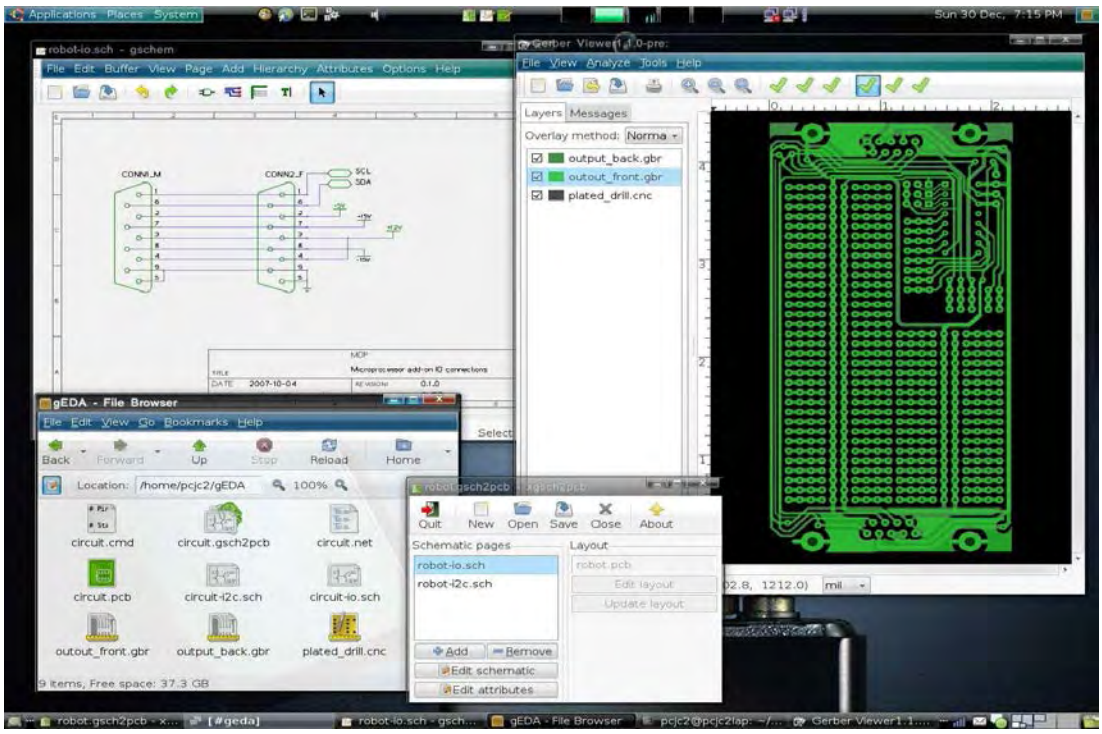
Electronic design automation (EDA or ECAD) is a category of software tools for designing electronic systems such as printed circuit boards and integrated circuits. The tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips. (wikipedia)

Picture 3.11



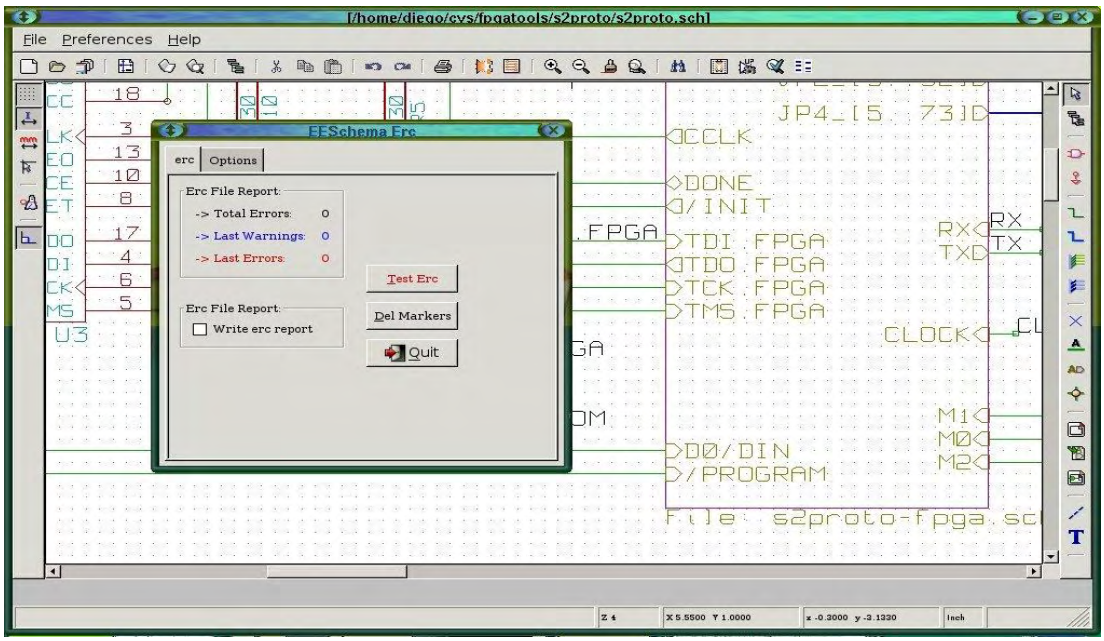
3D PCB layout

Picture 3.12



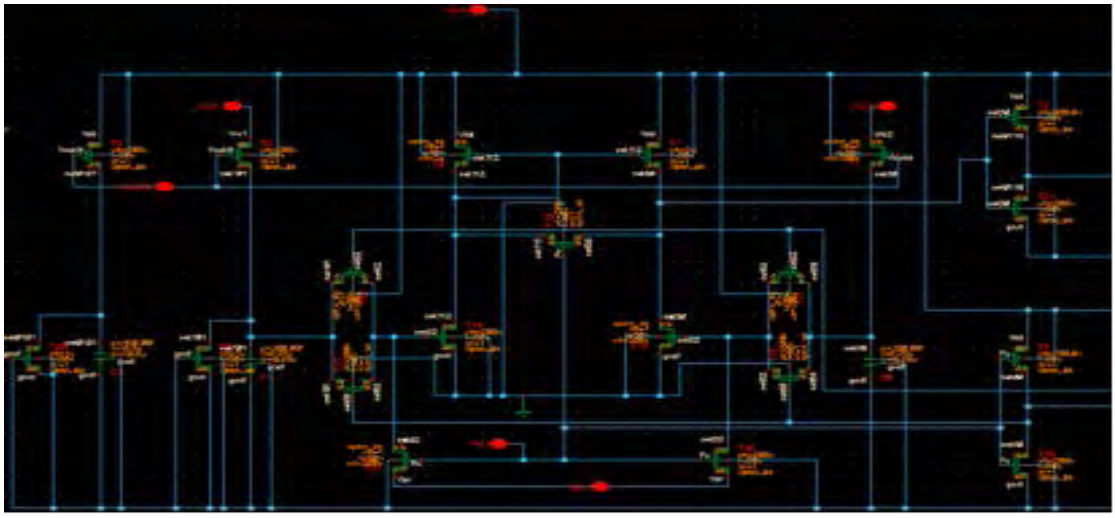
PCB layout and schematic for connector design

Picture 3.13



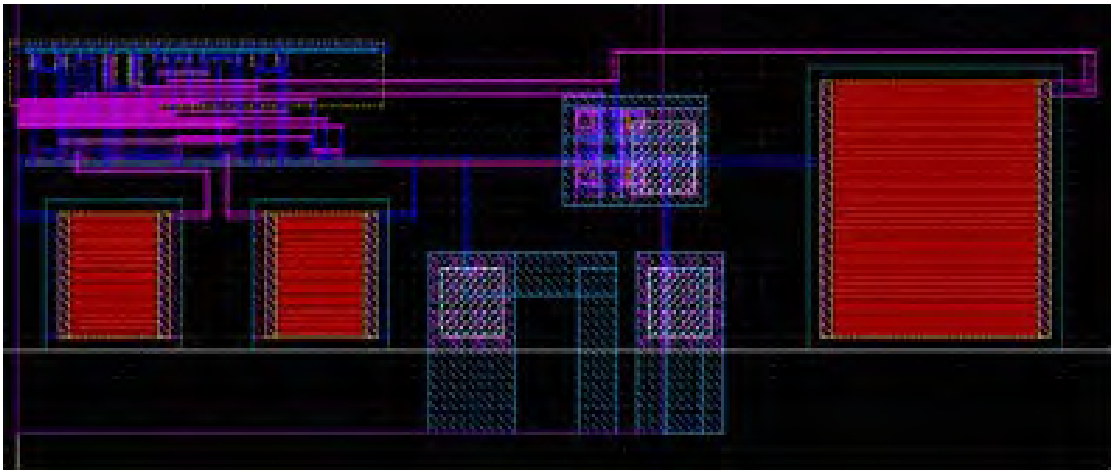
Schematic capture program

Picture 3.14



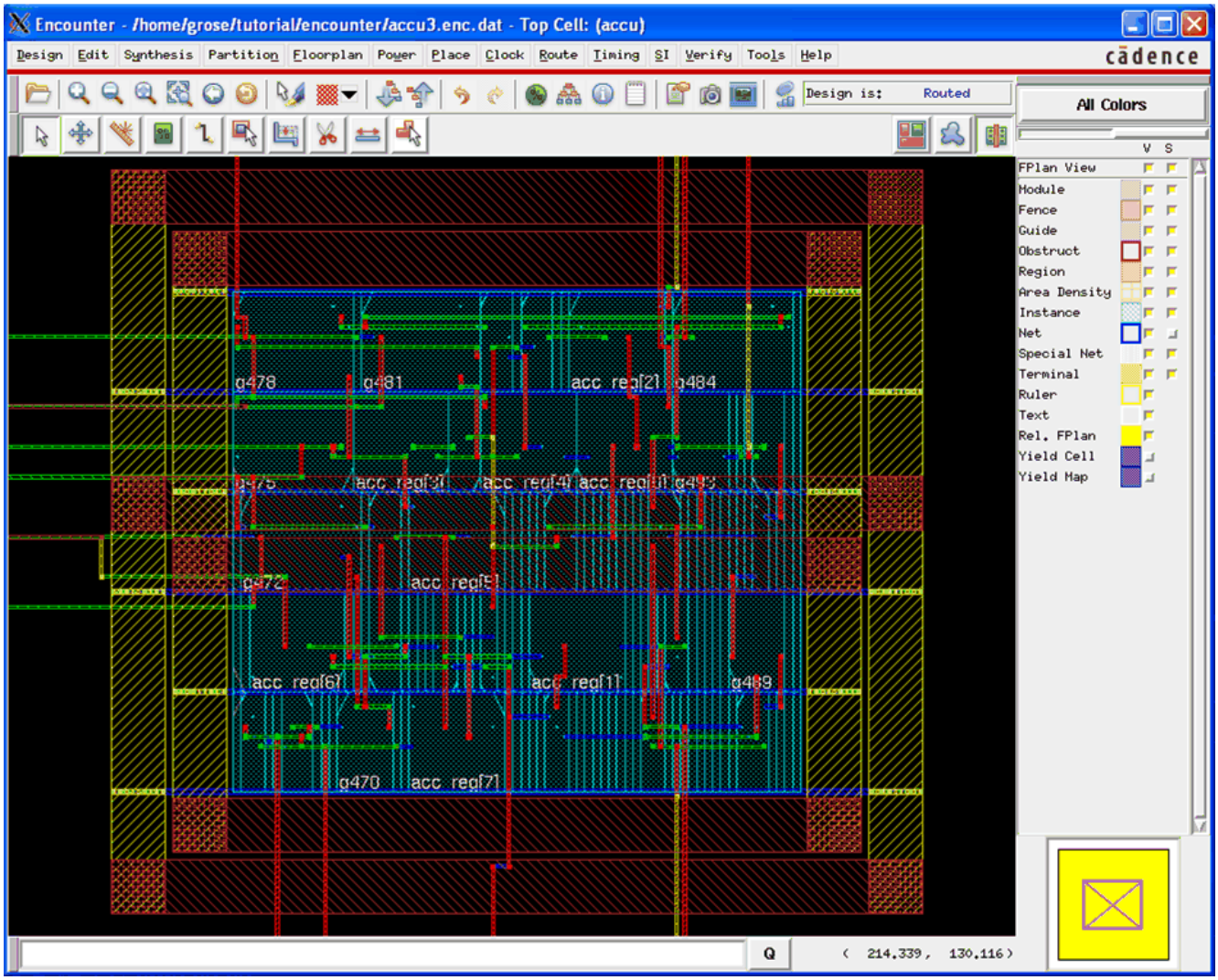
Design of Sense Amplifier circuit using Cadence Spectre tool.

Picture 3.15



Layout of charge sharing measurement circuit using the Cadence Virtuoso layout tool.

Picture 3.16



Cadence SOC Encounter

4. The placement problem definitions

4.1 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.

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

4.2 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.

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.

4.3 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.

4.4 Design Flow

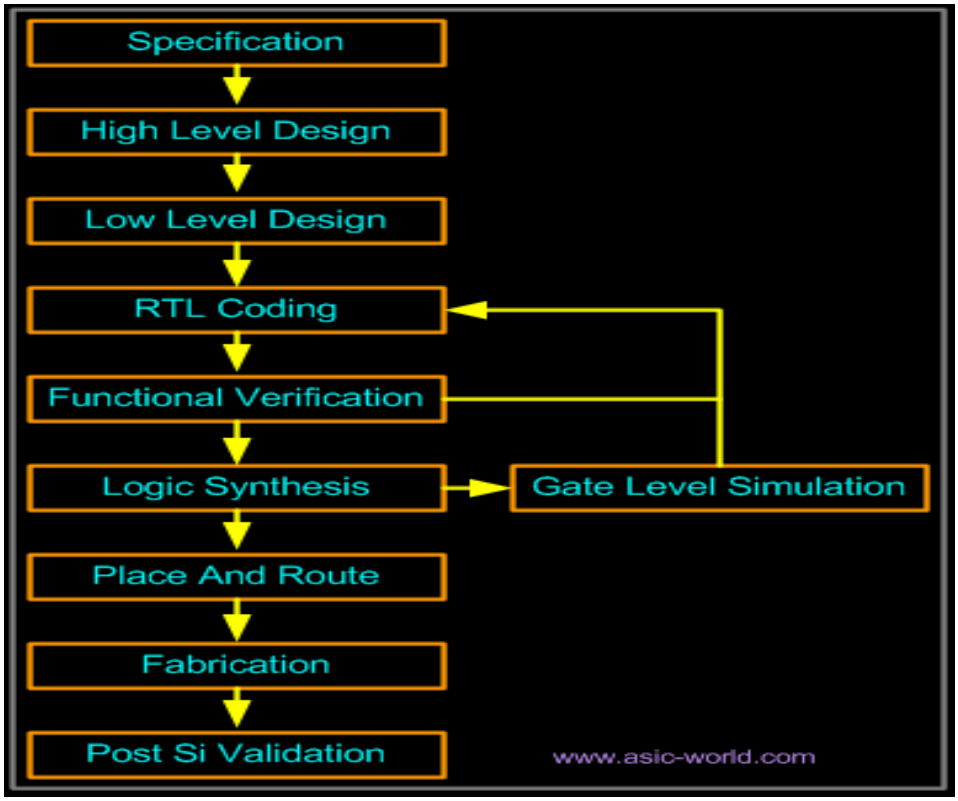
Design flows are the explicit combination of EDA tools to accomplish the design of an integrated circuit. Moore's Law has driven the entire IC implementation RTL to GDSII design flows from one which uses primarily standalone synthesis, placement, and routing algorithms to an integrated construction and 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. Picture 4.1, below, illustrates the ASIC flow.

Picture 4.1



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

4.4.1 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 4.1.

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]

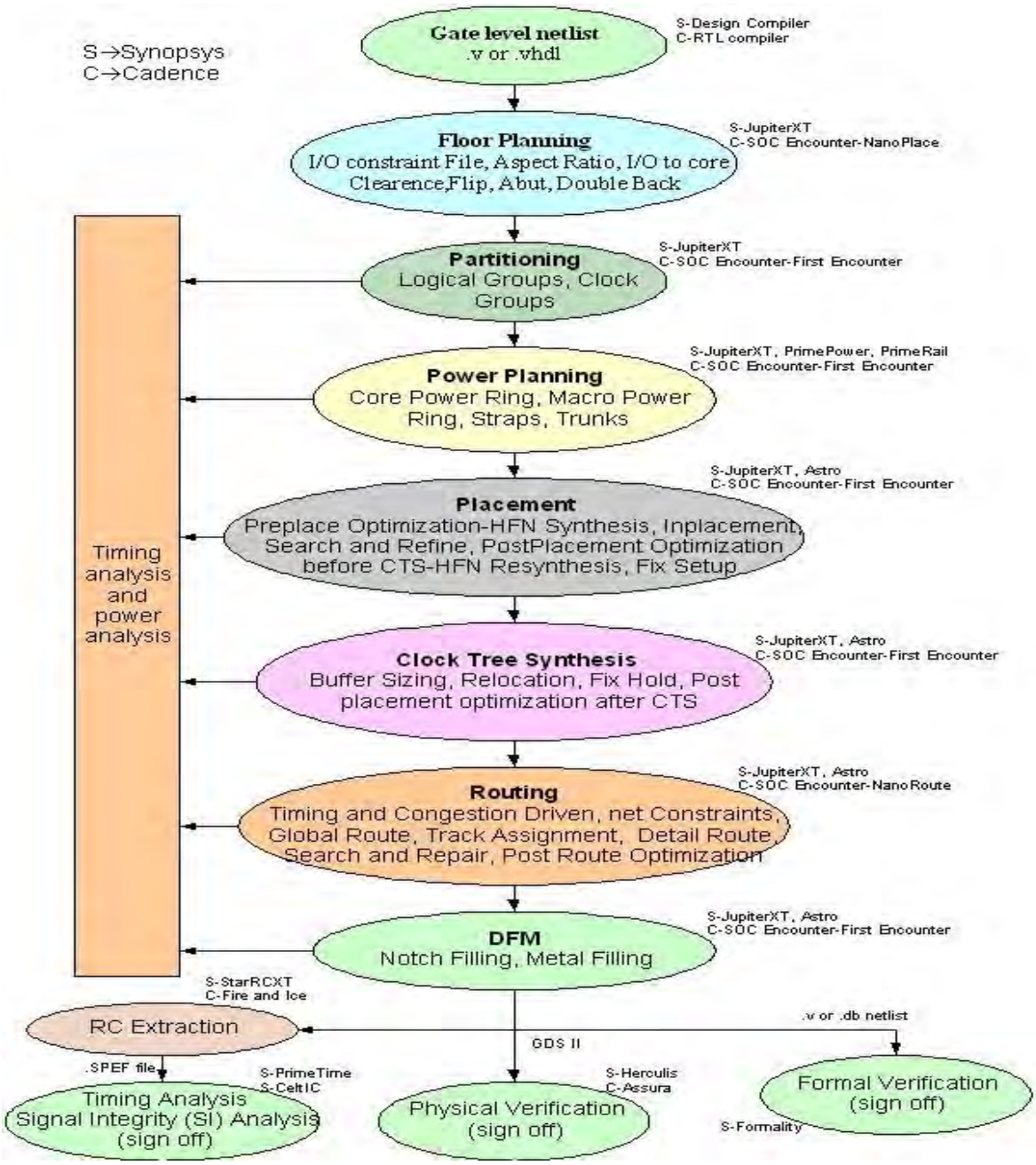
4.4.2 Back-end flow

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 picture 4.2.

Picture 4.2



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).

5. The visualization tool

This tool was conceived in order to represent with graphics the cell placement within a chip area. It uses data parsed from other tools like Cadence, and visualizes the data for an easy to understand image of the positions of the cells within the usable area. This makes it easier to assess the optimal positions of the cells, with colors distinguishing the unwanted overlapping cells and also a point of reference for the free space in the chip. Finally it provides information for each cell (type and size) and the option to export the full sized image of the chip to .png files.

5.1 File types that the tool uses

5.1.1. Input files for the algorithm

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

- 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 Output files that are used by the tool

The output of our algorithm is a .txt file that describes the final position of the cells of the IC. They are constructed in order to simplify the data inside the file and avoid working with VERILOG files, making it more accessible should one try to study them without a visualization tool.

The file structure is as follows:

- The dimensions file for each chip only has the height and width of the chip area. For example the dimension file for the S27 chip is this:

```
chip height = 5.600000 chip width = 5.538500
```

- The library size file for all the chips comprises of three columns. The type of the cell and the size of the cell in X and Y coordinates:

```
(std-cell Type)         X          Y

AND2_X1                0.760000 1.400000
AND2_X2                0.760000 1.400000
AOI211_X1              0.950000 1.400000
AOI211_X2              0.950000 1.400000
AOI211_X4              1.710000 1.400000
BUF_X1                 0.570000 1.400000
BUF_X16                1.140000 1.400000
INV_X1                 0.380000 1.400000
INV_X16                0.950000 1.400000
INV_X2                 0.380000 1.400000
```

- The legalized and unlegalized cell positions are described by the format in the respective files:

```
AOI21_X2    U26   3.662859   0.000000
NAND2_X2    U25   4.180000   2.800000
NAND3_X2    U24   0.950000   2.800000
NOR2_X2     U23   4.398846   4.200000
INV_X16     U22   0.028522   1.400000
INV_X8      U9    1.928515   1.400000
INV_X32     U8    2.660000   2.800000
Input       S1    5.538500   4.900000
input       S2    5.538500   3.500000
input       S3    5.538500   2.100000
output      S5    4.153875   5.600000
output      S7    4.153875   0.000000

etc…
```

5.2 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. These are the circuits that we are also going to use for demonstration and testing of the visualization tool.

5.3 Possible future upgrades and expansion ideas

- In its current state, the tool can only parse a specific file format. A major upgrade would be the ability to work with Cadence standard .lef and .def files without any format transformation prior to using them in the tool. More file formats could also be an option.
- A more user-friendly and efficient way to load the required files instead of manually choosing them one by one. This could also be upgraded to control the validity of the files loaded in order to avoid invalid results.
- A zoom in/zoom out option that could provide a better view of large chips areas.
- 3D rendering and animation abilities

- An extra module able to provide graphical representation of the connections (nets) between cells is also possible.
- Ability to assess qualitative and quantitative characteristics of circuits.
- Embed the algorithms developed by the VEDAlab and ability to work as a whole with them making this a complete EDA tool.

5.4 Technology, problems and solutions applied

This tool is built using Java. Libraries such as JGraphX were used to create vectors that would later allow interaction with the cells. JGraphX is a Java graph visualization library that can easily create graphs among other things.

Java tools such as Swing were also employed for the drawing of the graphics. The code is written in a way that makes it easy to understand and extend its functionality.

Problems that appeared during the development of the tool were mainly of algorithmic nature.

- ✓ The check whether two cells overlap each other for instance is implemented in the following way:

```
private String checkOverlap(Cell t, int i) {
        String color = "fillColor=green";
        Cell t1;
        for (int j = 0; j < cellList.size(); j++) {

            t1 = cellList.get(j);
            if (i == j || t1.getTypeName().equals("input") ||
t1.getTypeName().equals("output")) {
```

```
                continue;
            }
            if(t.rect.intersects(t1.rect)){
                color = "fillColor=red";
                overlapCounter++;
                break;
            }
        }
        return color;
    }
```

- ✓ Borders indicating the chip area are drawn using vertices from the JGraphX library.

```
 graph.setLabelsVisible(jCheckBox1flag);
        // draw vertices in order to create edges for borders
        Object v0,v1,v2,v3;
        v0 = graph.insertVertex(parent, "", null, 0 * scale, 0*scale, 0, 0);
        v1 = graph.insertVertex(parent, "", null, chipX * scale, 0*scale, 0, 0);
        v2 = graph.insertVertex(parent, "", null, 0 * scale, chipY*scale, 0, 0);
        v3 = graph.insertVertex(parent, "", null, chipX * scale, chipY*scale, 0, 0);
        graph.insertEdge(parent, "", null, v0,
v1,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
graph.insertEdge(parent, "", null, v1,
v3,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
        graph.insertEdge(parent, "", null, v3,
v2,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
        graph.insertEdge(parent, "", null, v2,
v0,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
```

- ✓ The parsing of the files is another important area of the tool development process. In this specific implementation I adapt the parser to the format of the input files by not taking into account the strings that are not needed. For example the chip dimensions are loaded as per the following code:

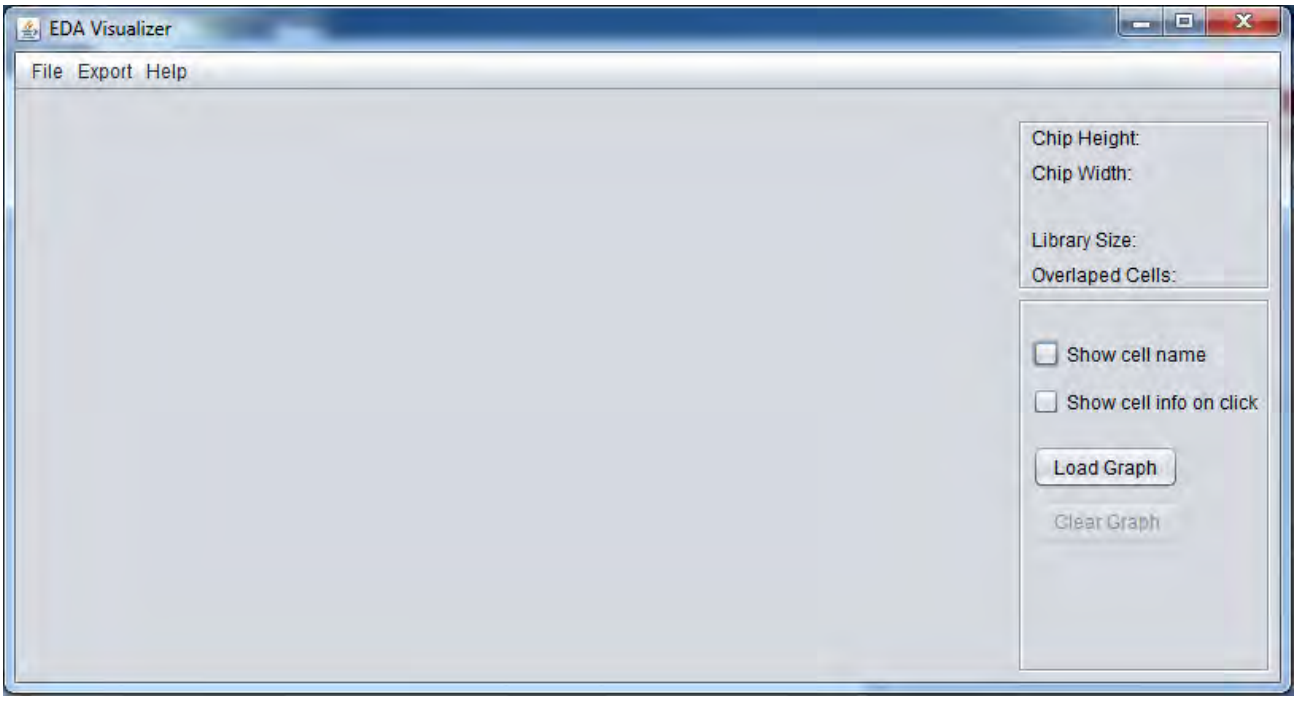
```
String hstr, wstr;
           try {
               Scanner sc = new Scanner(file);
               sc.next();
               sc.next();
               sc.next();
               hstr = sc.next();
               sc.next();
               sc.next();
               sc.next();
               wstr = sc.next();
               chipX = Float.parseFloat(wstr);
               chipY = Float.parseFloat(hstr);

               jLabelHeight.setText(hstr);
               jLabelWidth.setText(wstr);
           }
```

5.5 User guide to the tool

The tool is run via the executable .jar file. Java insures compatibility on multiple platforms provided that it (a Java package) is installed in the target machine.

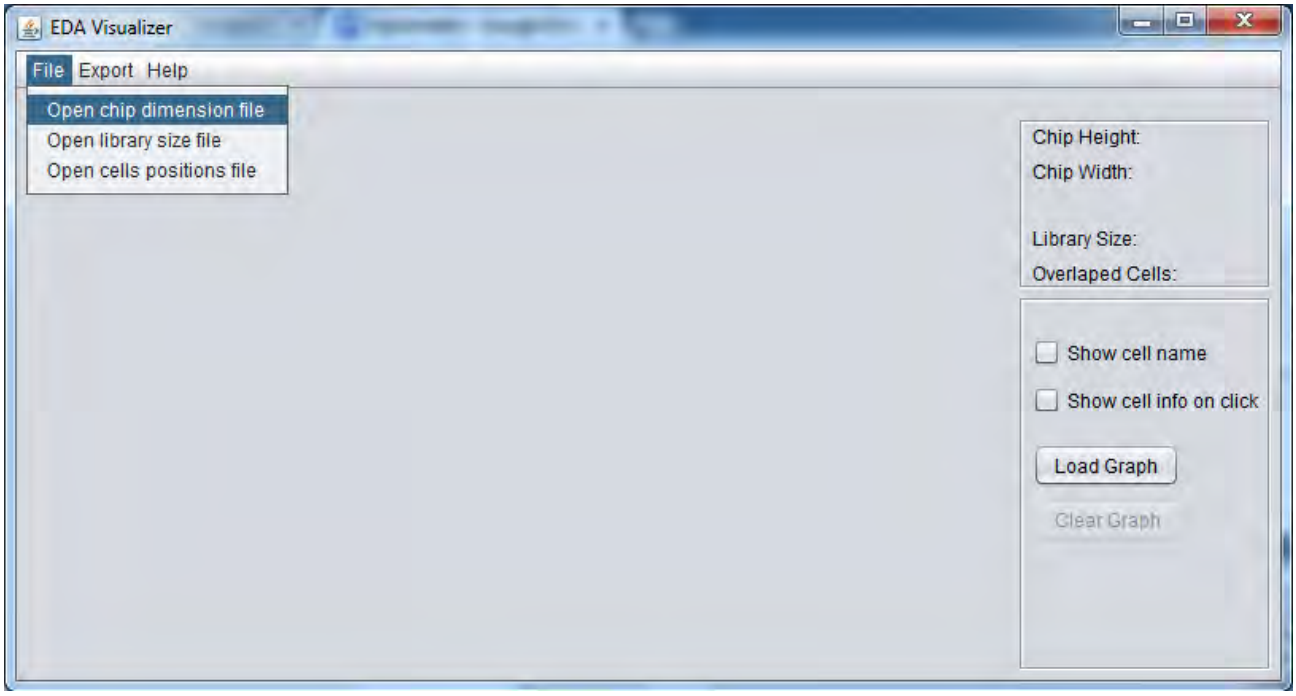
The window that the user is first greeted with is the following in the picture 5.1:

Picture 5.1



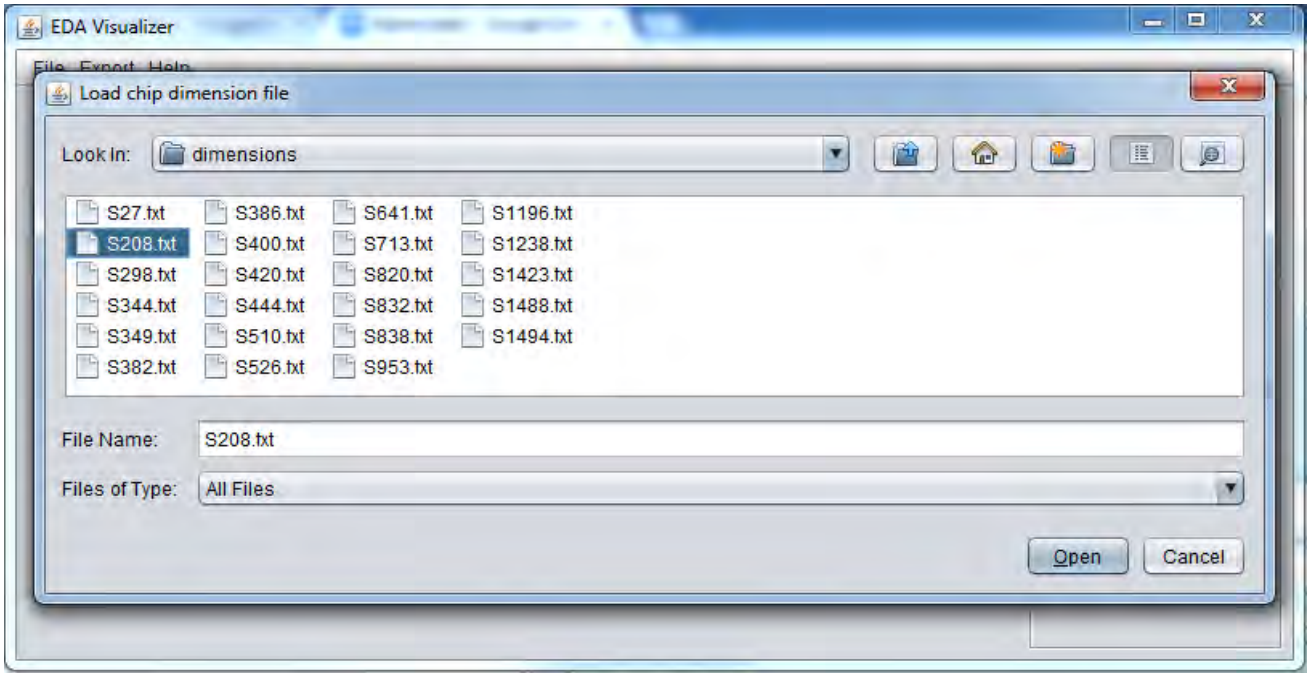
In here we can see a grey area that will draw the cells once it has all the required data and the “Load Graph” button is pressed. In order to start using the tool one must load the appropriate files in the following order. First one needs to open the “File” menu and select a chip dimensions file.

Picture 5.2



The next dialog gives the user the option to select the appropriate dimensions file:

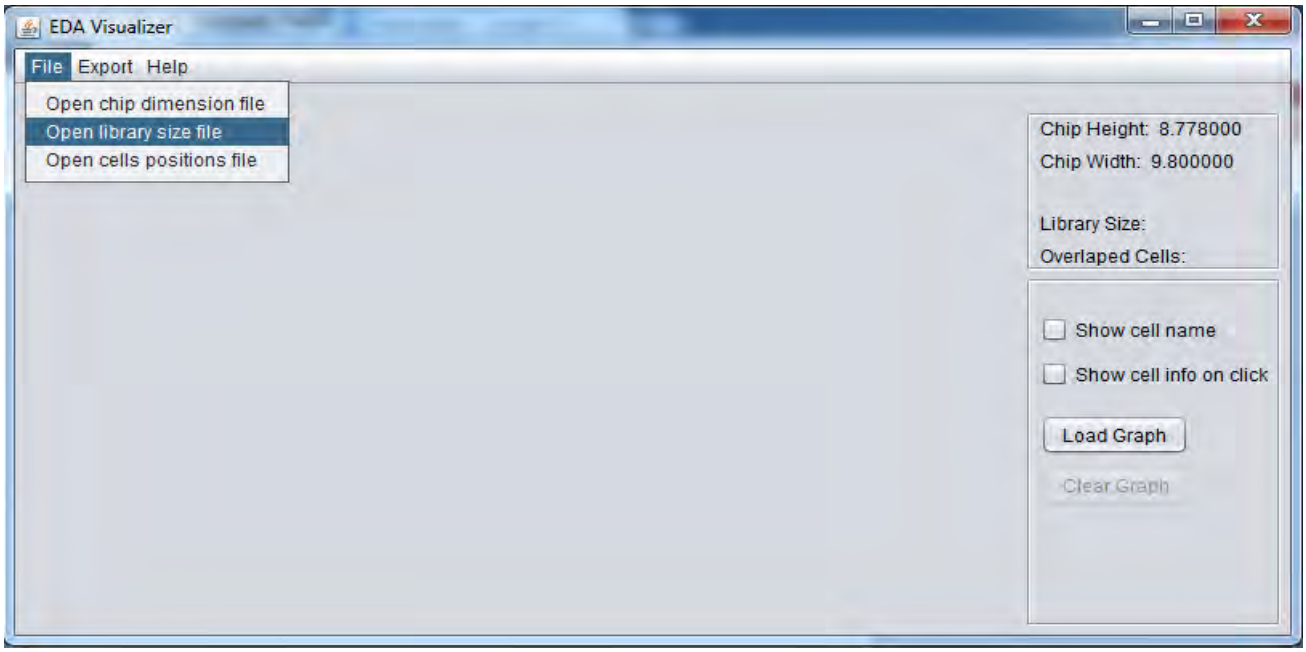
Picture 5.3



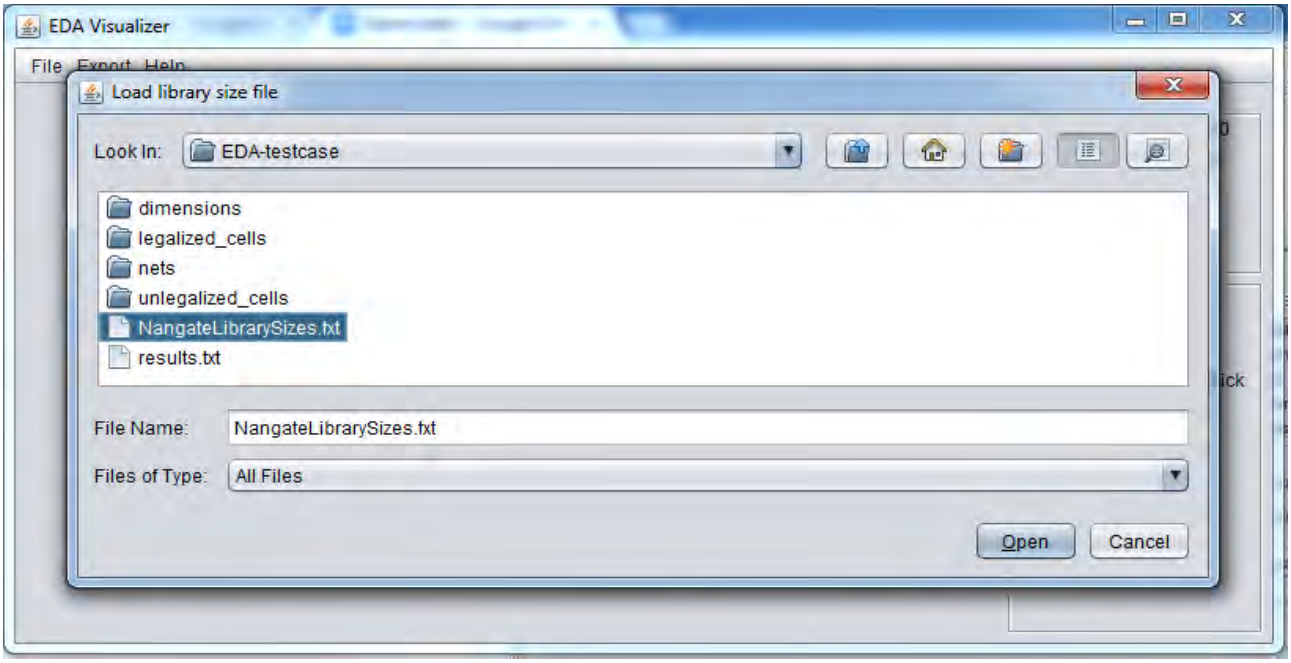
Now the user can see information about the chip dimensions.

Secondly the library size is required.

Picture 5.4



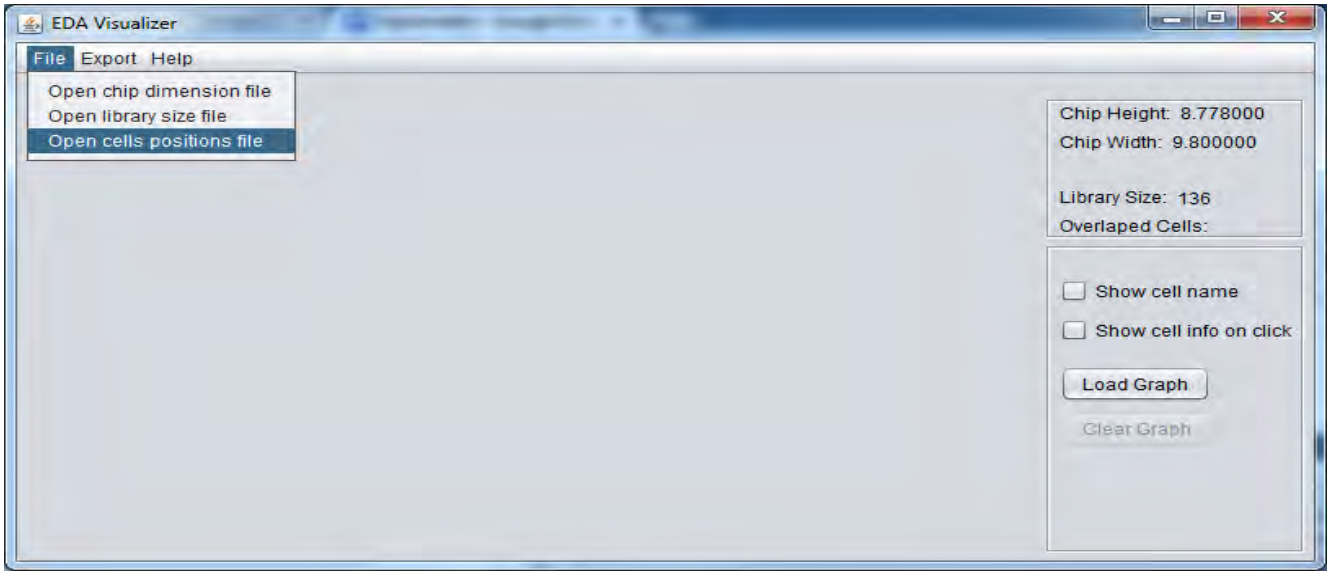
Picture 5.5



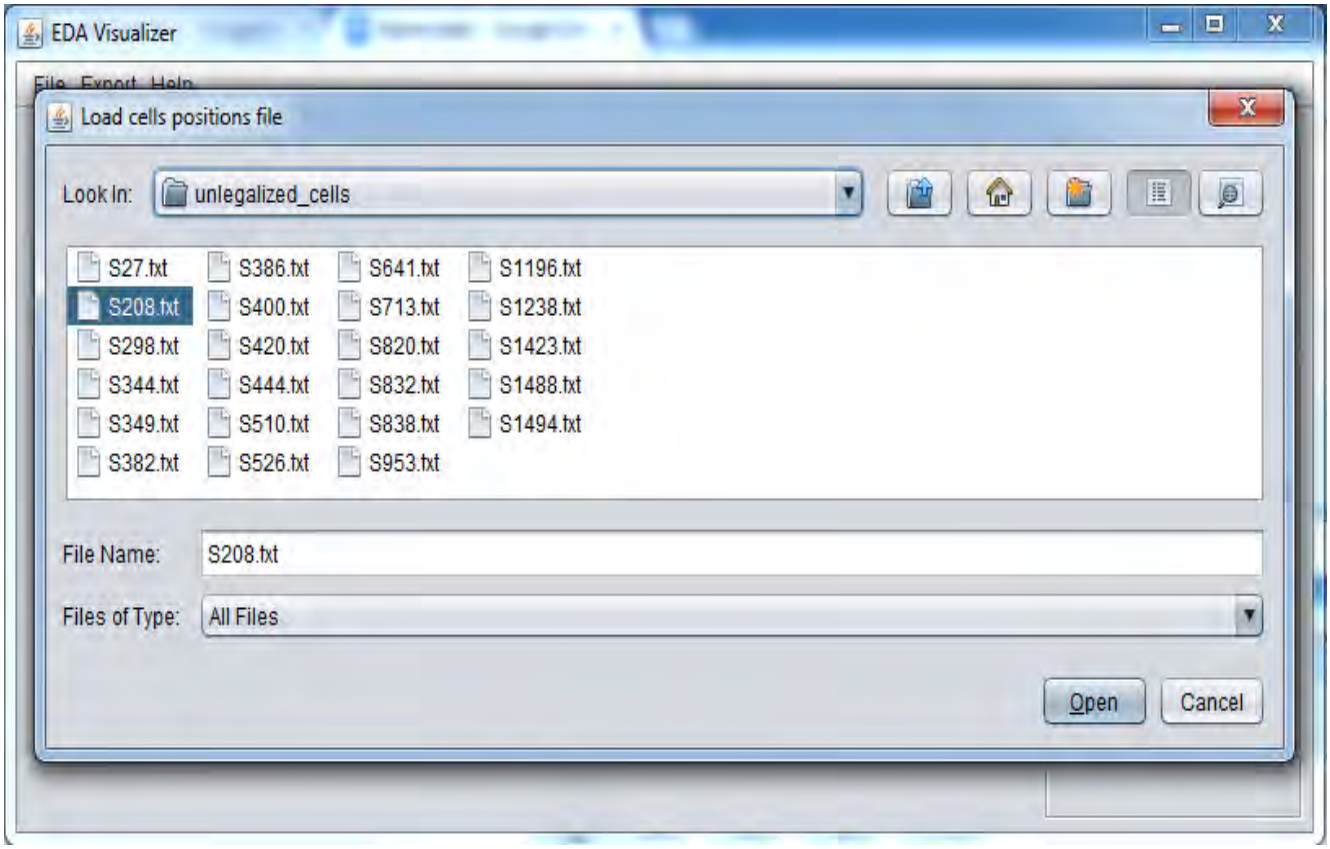
The library size appearing on the information box on the top right is the confirmation of successful parsing.

Thirdly the cells position file is loaded. This loads information about the location of each gate within the chip area.

Picture 5.6



Picture 5.7



Finally, after all three files are loaded into the tool, one needs to select “Load Graph”. This draws all the aforementioned information in an intuitive and simple to understand way.

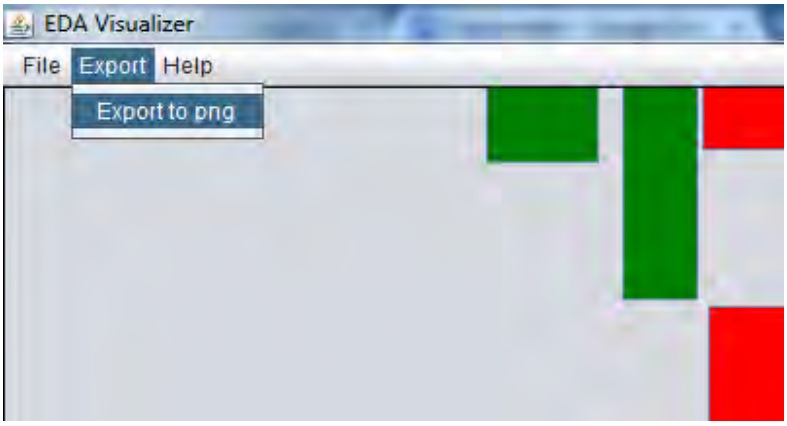
Picture 5.8

The image above shows the placement of one of the testbench chips. This particular example is the “unlegalized” positions of the cells. The overlapping cells are automatically counted and displayed on the top right box. They are drawn red to distinguish them from cells that do not overlap each other drawn in green.

The chip area will probably be larger than the usable area of the screen. The user can scroll around the rest of the chip area using the scroll bars below and on the right. Scrolling to the end of the chip area will reveal the squares colored black and blue. These represent the chip inputs and outputs respectively.

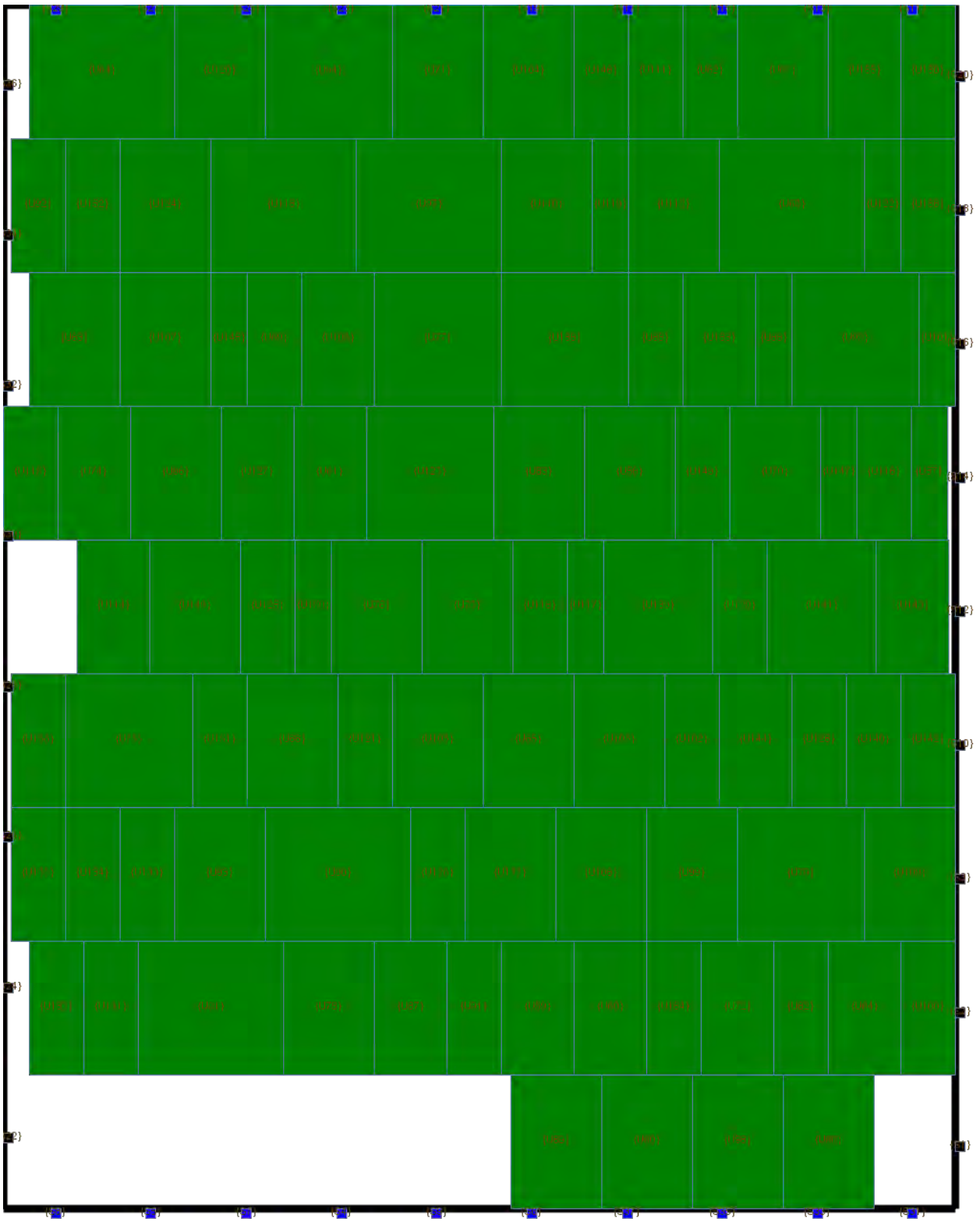
Using the “Export” menu one can export the whole chip image (not just the part shown within the scrollable area) to png format for a better overview. The .png file is named graph.png by default and is created in the same directory that the program is running.

Picture 5.9



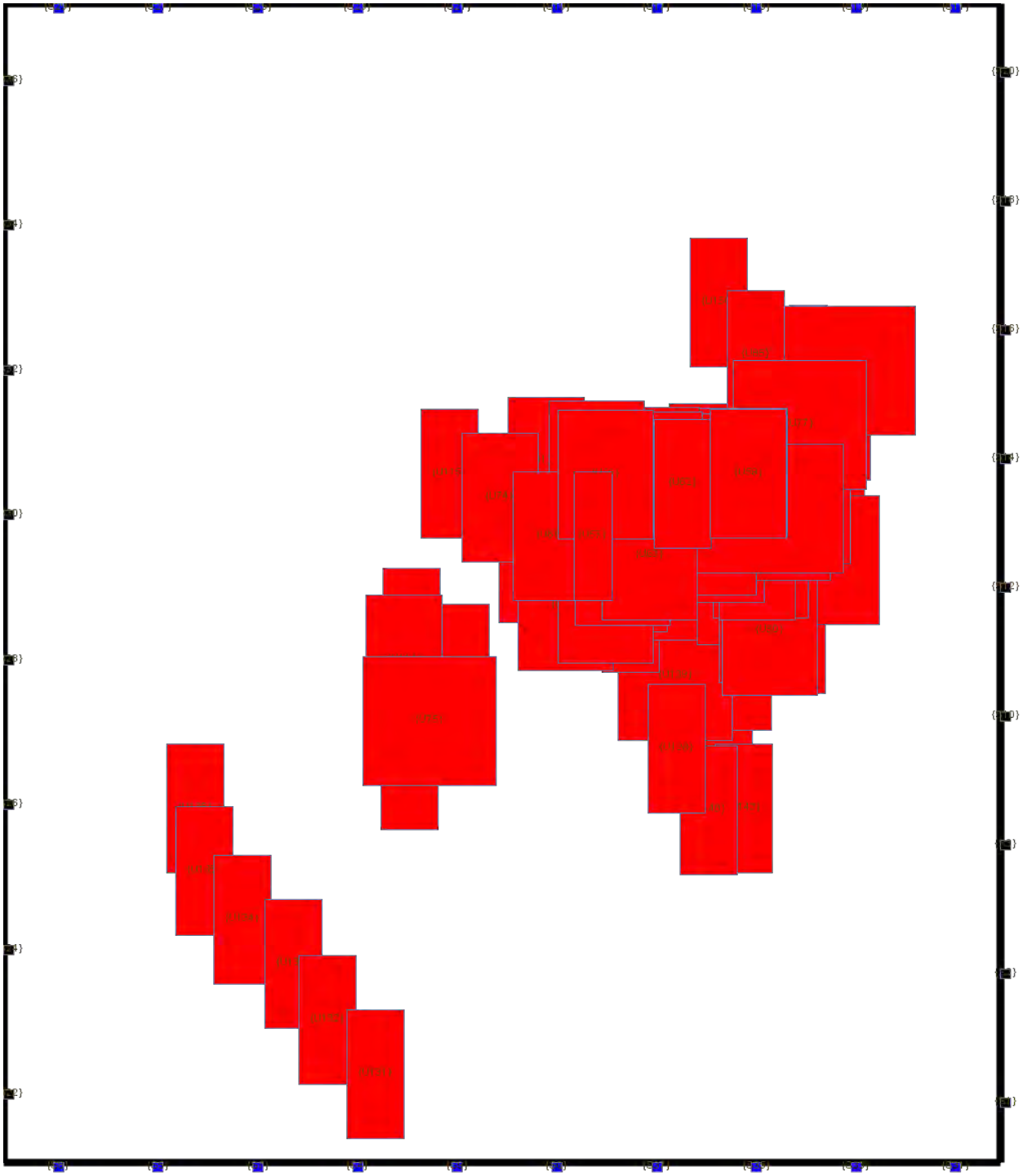
Here is an example of a chip with legalized cell positions. This shows all the cells in position, without any overlap. This is why they are all drawn green:

Picture 5.10



To better understand the difference, should one choose to display the “unlegalized” cell positions for the above chip, this would be the outcome:

Picture 5.11



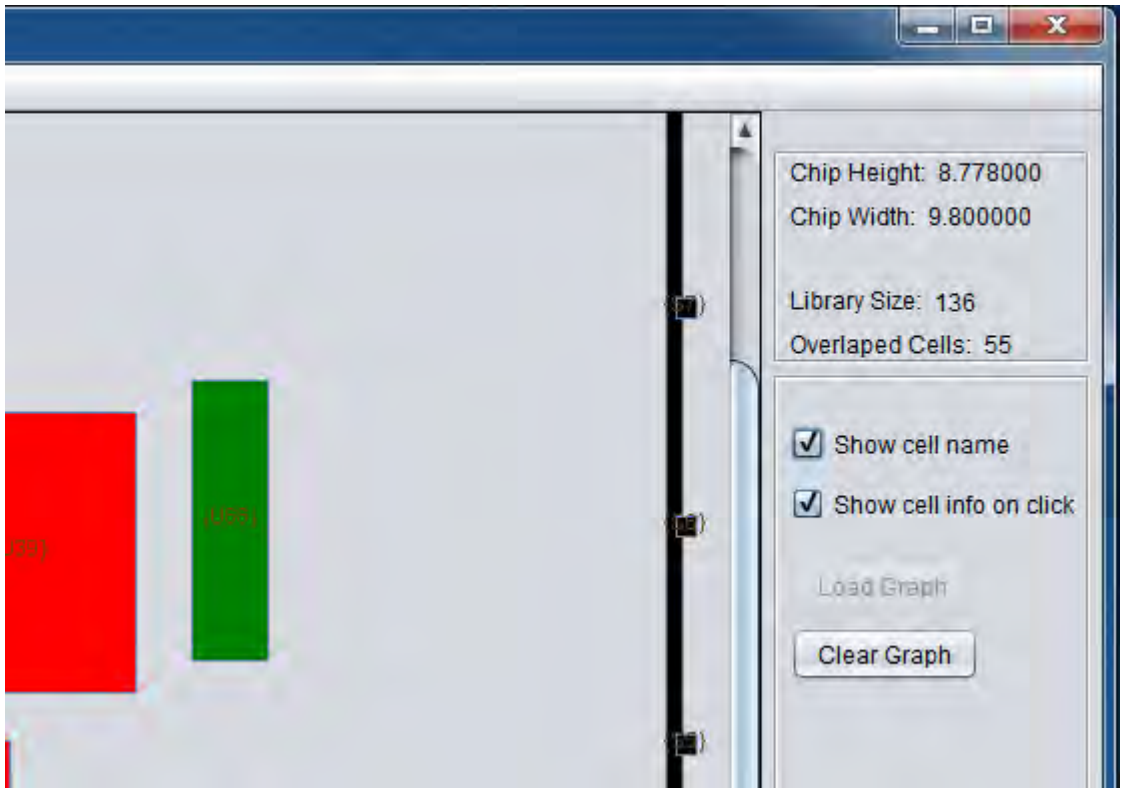
Extra care should be taken when loading the files. The user must be sure they are loading the corresponding cell positions file for the chip dimensions file they selected in

the first place. Failure to do so cannot be detected by the program so it will display erroneous data.

The options available once the graph is loaded are the following:

- “Clear Graph” which obviously clears the drawing area and resets the program so it is ready to load new files.
- “Show cell name” checkbox which toggles the names of the cells, visible or not.
- “Show cell info on click” which enables the user to click on each cell for further information.

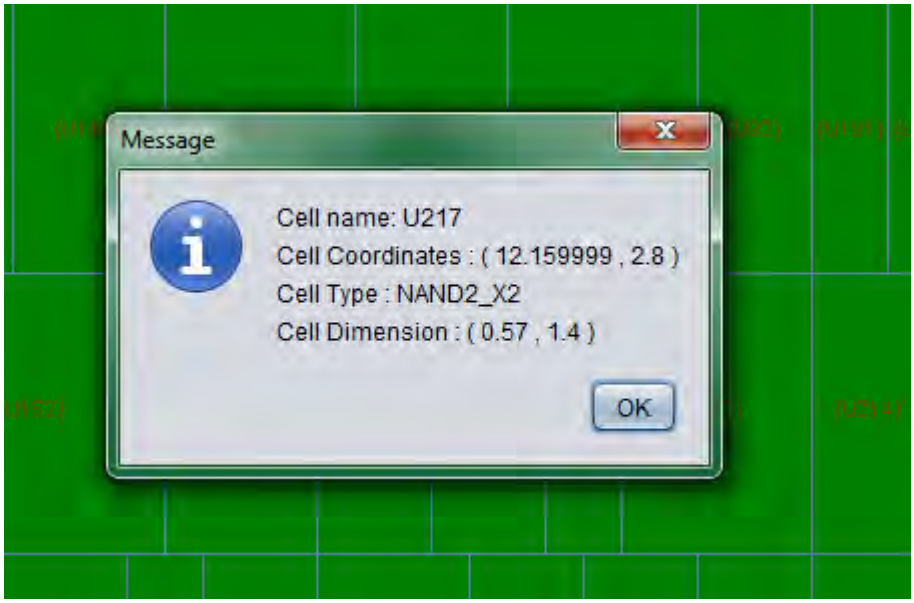
Picture 5.12



This is how it looks with the names enabled. Notice that the chip inputs and outputs have now names appearing as well.

If the user chooses to click on a cell, an input or an output node with the corresponding “info on click” option enabled the following window will appear.

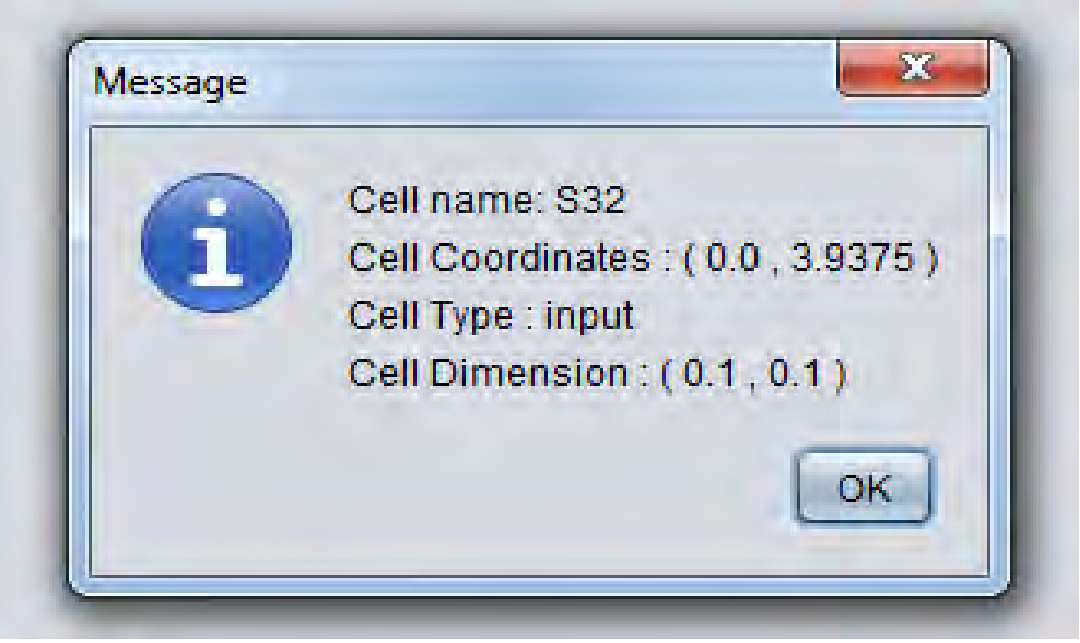
Picture 5.13



One can see the name of the selected cell, the position, the type of the gate and the inputs/outputs it drives as well as the dimensions of the cell.

This also works if one clicks on an input or output node. This is the information he will receive in that case:

Picture 5.14



6. Appendix

The following part contains all the Java source code for the visualization tool. Please note that all the automatically generated code from the Netbeans IDE will also be included. Parts of code that were automatically generated will be marked as such in order to avoid confusion.

The JGraphX library is also needed for compilation.

MainWindow.java

```
package chipsgui;

import com.mxgraph.model.mxCell;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.view.mxGraph;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics2D;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

/**
 *
 * @author Stergioulas George
 */
public class MainWindow extends javax.swing.JFrame {

    /**
```

```
     * Creates new form MainWindow
     */
    public final static boolean DEBUG = false;
    public final static String fileName = "S1494.txt";

    ArrayList<CellDim> cellDimList = new ArrayList<>();
    ArrayList<Cell> cellList = new ArrayList<>();
    ArrayList<Net> netList = new ArrayList<>(); //dimioyrgo 3 listes kai tis init
    double scale = 100;

    float chipY;
    float chipX;
    boolean jCheckBox1flag = false;
    boolean jCheckBox2flag = false;
    int overlapCounter=0;
    mxGraph graph;
    mxGraphComponent graphComponent;

    public MainWindow() {
        initComponents();
        jPanelGraph.setLayout(new BorderLayout());

    }

    private String checkOverlap(Cell t, int i) {
        String color = "fillColor=green";
        Cell t1;
        for (int j = 0; j < cellList.size(); j++) {

            t1 = cellList.get(j);
            if (i == j || t1.getTypeName().equals("input") ||
t1.getTypeName().equals("output")) {
                continue;
            }
            if(t.rect.intersects(t1.rect)){         //intersect einai etoimi
ylopoiisi tis rect tis java
                color = "fillColor=red";
                overlapCounter++;
                break;
            }
        }
        return color;
    }

   /* private Cell searchCell(String name) {
        int i;
        for (i = 0; i < cellList.size(); i++) {
            Cell t = cellList.get(i);
            if (name.equals(t.getName())) {
```

```
                return t;
            }
        }
        return null;
    } */

    private CellDim searchCellDim(String name) {
        int i;
        for (i = 0; i < cellDimList.size(); i++) {
            CellDim t = cellDimList.get(i);
            if (name.equals(t.getName())) {
                return t;
            }
        }
        return null;
    }

    private void createGraph() {
        Object parent = graph.getDefaultParent();

        graph.setEnabled(false);
        //graph.setConnectableEdges(false);
        graph.getModel().beginUpdate();

        graph.setLabelsVisible(jCheckBox1flag);
        // draw vertices in order to create edges for borders
        Object v0,v1,v2,v3;
        v0 = graph.insertVertex(parent, "", null, 0 * scale, 0*scale, 0, 0);
        v1 = graph.insertVertex(parent, "", null, chipX * scale, 0*scale, 0, 0);
        v2 = graph.insertVertex(parent, "", null, 0 * scale, chipY*scale, 0, 0);
        v3 = graph.insertVertex(parent, "", null, chipX * scale, chipY*scale, 0, 0);
        graph.insertEdge(parent, "", null, v0,
v1,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
        graph.insertEdge(parent, "", null, v1,
v3,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
        graph.insertEdge(parent, "", null, v3,
v2,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");
        graph.insertEdge(parent, "", null, v2,
v0,"startArrow=none;endArrow=none;strokeWidth=8;strokeColor=black");

        try {
            String color;
            Cell t;
            Object v;
            for (int i = 0; i < cellList.size(); i++) {
                t = cellList.get(i);
                if(t.getTypeName().equals("input")){
                    color = "fillColor=black";
                }else if(t.getTypeName().equals("output")){
```

```
                    color = "fillColor=blue";
                }else{
                    color = checkOverlap(t, i);
                }
                t.setColor(color);
                v = graph.insertVertex(parent, t.getName(), t, t.getX() * scale,
t.getY()*scale, t.getType().getXsize() * scale, t.getType().getYsize() * scale,
t.getColor());
            }
        } finally {
            graph.getModel().endUpdate();
            graphComponent.setSize((int)(chipX*scale), (int)(chipY*scale));
            graphComponent.setBorder(BorderFactory.createLineBorder(Color.black));
            graphComponent.setEnabled(false);

        }

        graphComponent.getGraphControl().addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {

                mxCell mx = (mxCell) graphComponent.getCellAt(e.getX(), e.getY());

                if(mx==null) {
                    return;
                }

                Cell cell = (Cell) mx.getValue();
                if (cell != null && jCheckBox2flag == true) {
                    JOptionPane.showMessageDialog(graphComponent, "Cell name: " +
cell.getName() + "\nCell Coordinates : ( " + cell.getX() + " , " + cell.getY() + "
)\nCell Type : " + cell.getType().getName() +"\nCell Dimension : (
"+cell.getType().getXsize()+" , "+cell.getType().getYsize()+" )");
                }
            }
        });

        jLabelBadCells.setText(Integer.toString(overlapCounter));
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        fileChooser = new javax.swing.JFileChooser();
        jCheckBoxMenuItem1 = new javax.swing.JCheckBoxMenuItem();
        jCheckBoxMenuItem2 = new javax.swing.JCheckBoxMenuItem();
        jCheckBoxMenuItem3 = new javax.swing.JCheckBoxMenuItem();
        jCheckBoxMenuItem4 = new javax.swing.JCheckBoxMenuItem();
```

```
jCheckBoxMenuItem5 = new javax.swing.JCheckBoxMenuItem();
buttonGroup1 = new javax.swing.ButtonGroup();
jFrame1 = new javax.swing.JFrame();
jPopupMenu1 = new javax.swing.JPopupMenu();
jPopupMenu2 = new javax.swing.JPopupMenu();
jPopupMenu3 = new javax.swing.JPopupMenu();
jCheckBoxMenuItem6 = new javax.swing.JCheckBoxMenuItem();
jPopupMenu4 = new javax.swing.JPopupMenu();
jPanel1 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jLabelWidth = new javax.swing.JLabel();
jLabelHeight = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabelLibrarySize = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabelBadCells = new javax.swing.JLabel();
jScrollPane1 = new javax.swing.JScrollPane();
jPanelGraph = new javax.swing.JPanel();
jTextField2 = new javax.swing.JTextField();
jTextField3 = new javax.swing.JTextField();
jPanel2 = new javax.swing.JPanel();
jCheckBox1 = new javax.swing.JCheckBox();
jCheckBox2 = new javax.swing.JCheckBox();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
jMenu1 = new javax.swing.JMenu();
jMenuItemChipDim = new javax.swing.JMenuItem();
jMenuItemCellDim = new javax.swing.JMenuItem();
jMenuItemCells = new javax.swing.JMenuItem();
jMenu2 = new javax.swing.JMenu();
jMenuItem1 = new javax.swing.JMenuItem();
jMenu3 = new javax.swing.JMenu();
jMenuItem3 = new javax.swing.JMenuItem();

jCheckBoxMenuItem1.setSelected(true);
jCheckBoxMenuItem1.setText("jCheckBoxMenuItem1");

jCheckBoxMenuItem2.setSelected(true);
jCheckBoxMenuItem2.setText("jCheckBoxMenuItem2");

jCheckBoxMenuItem3.setSelected(true);
jCheckBoxMenuItem3.setText("jCheckBoxMenuItem3");

jCheckBoxMenuItem4.setSelected(true);
jCheckBoxMenuItem4.setText("jCheckBoxMenuItem4");

jCheckBoxMenuItem5.setSelected(true);
```

```
        jCheckBoxMenuItem5.setText("jCheckBoxMenuItem5");

        javax.swing.GroupLayout jFrame1Layout = new
javax.swing.GroupLayout(jFrame1.getContentPane());
        jFrame1.getContentPane().setLayout(jFrame1Layout);
        jFrame1Layout.setHorizontalGroup(

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 400, Short.MAX_VALUE)
        );
        jFrame1Layout.setVerticalGroup(

jFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 300, Short.MAX_VALUE)
        );

        jCheckBoxMenuItem6.setSelected(true);
        jCheckBoxMenuItem6.setText("jCheckBoxMenuItem6");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("EDA Visualizer");
        setName("EDA Visualizer"); // NOI18N

        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        jPanel1.setToolTipText("Cell dimensions");

        jLabel1.setText("Chip Height:");

        jLabel2.setText("Chip Width:");

        jLabel5.setText("Library Size:");

        jLabel4.setText("Overlaped Cells:");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                        .addComponent(jLabelHeight,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabelWidth,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabelBadCells,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jLabelLibrarySize,
javax.swing.GroupLayout.PREFERRED_SIZE, 51, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 19, Short.MAX_VALUE)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                    .addComponent(jLabelWidth, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
, false)
                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabelHeight, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGap(26, 26, 26)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
```

```
                    .addComponent(jLabel5,
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabelLibrarySize,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED_SIZE,
14, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jLabel4)
                    .addComponent(jLabelBadCells,
javax.swing.GroupLayout.PREFERRED_SIZE, 14, javax.swing.GroupLayout.PREFERRED_SIZE)))
        );

        jScrollPane1.setBorder(null);

        jPanelGraph.setPreferredSize(new java.awt.Dimension(0, 0));

        javax.swing.GroupLayout jPanelGraphLayout = new
javax.swing.GroupLayout(jPanelGraph);
        jPanelGraph.setLayout(jPanelGraphLayout);
        jPanelGraphLayout.setHorizontalGroup(

jPanelGraphLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 625, Short.MAX_VALUE)
        );
        jPanelGraphLayout.setVerticalGroup(

jPanelGraphLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 360, Short.MAX_VALUE)
        );

        jScrollPane1.setViewportView(jPanelGraph);

        jTextField2.setText("jTextField2");

        jTextField3.setText("jTextField3");

        jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jCheckBox1.setText("Show cell name");
        jCheckBox1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jCheckBox1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jCheckBox1ActionPerformed(evt);
            }
        });

        jCheckBox2.setText("Show cell info on click");
```

```
        jCheckBox2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jCheckBox2ActionPerformed(evt);
            }
        });

        jButton1.setText("Load Graph");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Clear Graph");
        jButton2.setEnabled(false);
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
)
                    .addComponent(jButton2)
                    .addComponent(jCheckBox1)
                    .addComponent(jCheckBox2)
                    .addComponent(jButton1))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addContainerGap(23, Short.MAX_VALUE)
                .addComponent(jCheckBox1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jCheckBox2)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
```

```
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton2)
        .addGap(79, 79, 79))
);

jCheckBox1.getAccessibleContext().setAccessibleName("Show cell names");

jMenu1.setText("File");

jMenuItemChipDim.setText("Open chip dimension file");
jMenuItemChipDim.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemChipDimActionPerformed(evt);
    }
});
jMenu1.add(jMenuItemChipDim);

jMenuItemCellDim.setText("Open library size file");
jMenuItemCellDim.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemCellDimActionPerformed(evt);
    }
});
jMenu1.add(jMenuItemCellDim);

jMenuItemCells.setText("Open cells positions file");
jMenuItemCells.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItemCellsActionPerformed(evt);
    }
});
jMenu1.add(jMenuItemCells);

jMenuBar1.add(jMenu1);

jMenu2.setText("Export");

jMenuItem1.setText("Export to png");
jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem1ActionPerformed(evt);
    }
});
jMenu2.add(jMenuItem1);

jMenuBar1.add(jMenu2);

jMenu3.setText("Help");
jMenu3.addActionListener(new java.awt.event.ActionListener() {
```

```
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenu3ActionPerformed(evt);
            }
        });

        jMenuItem3.setText("Info");
        jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem3ActionPerformed(evt);
            }
        });
        jMenu3.add(jMenuItem3);

        jMenuBar1.add(jMenu3);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
625, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jScrollPane1)
            .addGroup(layout.createSequentialGroup()
                .addGap(20, 20, 20)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
        jPanel1.getAccessibleContext().setAccessibleName("");

        pack();
    }// </editor-fold>

    private void jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
        graph.getModel().beginUpdate();
        jCheckBox1flag = !jCheckBox1flag;
        graph.setLabelsVisible(jCheckBox1flag);
        graph.refresh();
        graph.getModel().endUpdate();
    }

    private void jCheckBox2ActionPerformed(java.awt.event.ActionEvent evt) {

        if (jCheckBox2.isSelected() == true) {
            jCheckBox2flag = true;
        } else {
            jCheckBox2flag = false;
        }

    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        Dimension d = graphComponent.getGraphControl().getSize();
        BufferedImage image = new BufferedImage(d.width, d.height,
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g = image.createGraphics();
        graphComponent.getGraphControl().paint(g);
        final File outputfile = new File("graph.png");
        try {
            ImageIO.write(image, "png", outputfile);
        } catch (IOException ex) {
            Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    private void jMenuItemCellsActionPerformed(java.awt.event.ActionEvent evt) {
        cellList.clear();
        fileChooser.setDialogTitle("Load cells positions file");
        int returnVal;
        if(DEBUG) {
            returnVal = JFileChooser.APPROVE_OPTION;
        }
        else {
            returnVal = fileChooser.showOpenDialog(this);
        }
```

```
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file;
            if(DEBUG) {
                file = new File("EDA-testcase\\legalized_cells\\"+fileName);
            }
            else {
                file = fileChooser.getSelectedFile();
            }

            try {
                Scanner sc = new Scanner(file);

                while (sc.hasNext()) {
                    String typename = sc.next();
                    String name = sc.next();            //Cell name
                    Float x = Float.valueOf(sc.next()); //x-coord
                    Float y = Float.valueOf(sc.next()); // y-coord
                    if (typename.contains("input") || typename.contains("output")) {
                        //continue;
                    }

                    cellList.add(new Cell(typename, name, x, y));
                }
//                jLabelCellsCounter.setText(Integer.toString(cellList.size()));
            } catch (FileNotFoundException ex) {
                Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null,
ex);
            }
        } else {
            System.out.println("File access cancelled by user.");
        }
    }

    private void jMenuItemCellDimActionPerformed(java.awt.event.ActionEvent evt) {
        cellDimList.clear();
        fileChooser.setDialogTitle("Load library size file");
        int returnVal;
        if(DEBUG) {
            returnVal = JFileChooser.APPROVE_OPTION;
        }
        else {
            returnVal = fileChooser.showOpenDialog(this);
        }

        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file;
            if(DEBUG)
                file = new File("EDA-testcase\\NangateLibrarySizes.txt");
```

```
            else
                file = fileChooser.getSelectedFile();


            try {
                Scanner sc = new Scanner(file);
                String name;
                float xsize, ysize;
                sc.next();
                sc.next();
                sc.next();
                sc.next();
                while (sc.hasNext()) {
                    name = sc.next();
                    xsize = Float.valueOf(sc.next());
                    ysize = Float.valueOf(sc.next());
                    cellDimList.add(new CellDim(name, xsize, ysize));
                }
                cellDimList.add(new CellDim("input", (float)0.1, (float)0.1));
                cellDimList.add(new CellDim("output", (float)0.1, (float)0.1));
                jLabelLibrarySize.setText(Integer.toString(cellDimList.size()));

            } catch (FileNotFoundException ex) {
                Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null,
ex);
            } catch (InputMismatchException ex) {

            }
        }
    }

    private void jMenuItemChipDimActionPerformed(java.awt.event.ActionEvent evt) {
        fileChooser.setDialogTitle("Load chip dimension file");
        int returnVal;
        if(DEBUG) {
            returnVal = JFileChooser.APPROVE_OPTION;
        }
        else {
            returnVal = fileChooser.showOpenDialog(this);
        }
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file;
            if(DEBUG) {
                file = new File("EDA-testcase\\dimensions\\"+fileName);
            }
            else {
                file = fileChooser.getSelectedFile();
            }
```

```
            String hstr, wstr;
            try {
                Scanner sc = new Scanner(file);
                sc.next();
                sc.next();
                sc.next();
                hstr = sc.next();
                sc.next();
                sc.next();
                sc.next();
                wstr = sc.next();
                chipX = Float.parseFloat(wstr);
                chipY = Float.parseFloat(hstr);

                jLabelHeight.setText(hstr);
                jLabelWidth.setText(wstr);
            } catch (FileNotFoundException ex) {
                Logger.getLogger(MainWindow.class.getName()).log(Level.SEVERE, null,
ex);
            }
        } else {
            System.out.println("File access cancelled by user.");
        }
    }

    private void jMenu3ActionPerformed(java.awt.event.ActionEvent evt) {
    }

    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(graphComponent, "Author : George Stergioulas" +
"\n" + "Department of Computer, Communications and Network Engineering
"+"\n"+"University of Thessaly"+"\n"+"2012-2013"+"\n"+"Copyright © All Rights
Reserved");
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //jPanelGraph.add(graphComponent);
        if(DEBUG){
            jMenuItemCellDim.doClick();
            jMenuItemChipDim.doClick();
            jMenuItemCells.doClick();
        }
        boolean flag=false;
        if(cellDimList.isEmpty()){
            JOptionPane.showMessageDialog(this, "Cells dimensions not loaded", "Cell
Dimensions Error", JOptionPane.ERROR_MESSAGE);
            flag = true;
        }
        if(cellList.isEmpty()){
```

```
            JOptionPane.showMessageDialog(this, "Cells positions not loaded", "Cell
Position Error", JOptionPane.ERROR_MESSAGE);
            flag = true;
        }

        if(netList.isEmpty()){
            //JOptionPane.showMessageDialog(this, "Netlist not loaded", "Netlist
Error", JOptionPane.ERROR_MESSAGE);
            //flag = true;
        }
        if(flag) return;

        int i;
        CellDim type;
        for(i=0;i<cellList.size();i++){
            Cell cell = cellList.get(i);
            type = searchCellDim(cell.getTypeName());
            if(type==null) continue;
            cell.setType(type);
        }
        graph = new mxGraph();
        graphComponent = new mxGraphComponent(graph);
        createGraph();


        jPanelGraph.add(graphComponent);
        jPanelGraph.revalidate();
        jButton1.setEnabled(false);
        jButton2.setEnabled(true);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        cellDimList.clear();
        cellList.clear();
        netList.clear();

        graphComponent.removeAll();

        jPanelGraph.removeAll();
        jPanelGraph.revalidate();
        jPanelGraph.repaint();

        jButton1.setEnabled(true);
        jButton2.setEnabled(false);

        overlapCounter = 0;

        jLabelBadCells.setText("");
```

```
        //jLabelCellsCounter.setText("");
        jLabelHeight.setText("");
        jLabelLibrarySize.setText("");
        jLabelWidth.setText("");
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException | InstantiationException |
IllegalAccessException | javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainWindow.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                MainWindow w = new MainWindow();
                w.setVisible(true);
            }
        });

    }
    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JFileChooser fileChooser;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JCheckBox jCheckBox1;
```

```
    private javax.swing.JCheckBox jCheckBox2;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem1;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem2;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem3;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem4;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem5;
    private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem6;
    private javax.swing.JFrame jFrame1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabelBadCells;
    private javax.swing.JLabel jLabelHeight;
    private javax.swing.JLabel jLabelLibrarySize;
    private javax.swing.JLabel jLabelWidth;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenu jMenu3;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JMenuItem jMenuItemCellDim;
    private javax.swing.JMenuItem jMenuItemCells;
    private javax.swing.JMenuItem jMenuItemChipDim;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanelGraph;
    private javax.swing.JPopupMenu jPopupMenu1;
    private javax.swing.JPopupMenu jPopupMenu2;
    private javax.swing.JPopupMenu jPopupMenu3;
    private javax.swing.JPopupMenu jPopupMenu4;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    // End of variables declaration
}
```

Cell.java

```
package chipsgui;

import java.awt.geom.Rectangle2D;

/**
```

```
 *
 * @author George Stergioulas
 */
public class Cell {
    private CellDim type;
    private String name;
    private String typename;
    private String color;
    private float x;
    private float y;
    private final float TOL = (float)0.0001;
    public Rectangle2D rect;

    public Cell(CellDim type, String name, float x, float y) {
        this.type = type;
        this.name = name;
        this.x = x;
        this.y = y;
        rect = new Rectangle2D.Float(x, y, type.getXsize()-TOL, type.getYsize()-TOL);
    }

    public Cell(String typename, String name, float x, float y) {
        this.typename = typename;
        this.name = name;
        this.x = x;
        this.y = y;
    }

    public CellDim getType() {
        return type;
    }



    public void setType(CellDim type) {
        this.type = type;
        rect = new Rectangle2D.Float(x, y, type.getXsize()-TOL, type.getYsize()-TOL);
    }

     public String getTypeName() {
        return typename;
    }
     public void setTypeName(String typename){
         this.typename = typename;
     }
    public String getName() {
        return name;
    }
     public String getColor() {
        return color;
```

```
    }

     public void setColor(String color) {
        this.color = color;
    }
    public void setName(String name) {
        this.name = name;
    }

    public float getX() {
        return x;
    }

    public void setX(float x) {
        this.x = x;
    }

    public float getY() {
        return y;
    }

    public void setY(float y) {
        this.y = y;

    }

    @Override
    public String toString() {
        return '{' + name + '}';
    }

}
```

CellDim.java

```
package chipsgui;

/**
 *
```

```
 * @author George Stergioulas
 */
class CellDim {
    private String name;
    private float xsize;
    private float ysize;

    public CellDim(String name, float x, float y) {
        this.name = name;
        this.xsize = x;
        this.ysize = y;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public float getXsize() {
        return xsize;
    }

    public void setXsize(float xsize) {
        this.xsize = xsize;
    }

    public float getYsize() {
        return ysize;
    }

    public void setYsize(float ysize) {
        this.ysize = ysize;
    }



}
```

7. References

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

2. 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.

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

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

5. 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.

6. 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.

7. 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.

8. Walter Savitch, Kenrick Mock. Absolute Java, 5th Edition. Addison-Wesley 2012.

9. David Etheridge. Java: Graphical User Interfaces - An Introduction to Java Programming. Bookboon 2009.

10. Kathy Sierra, Bert Bates. Head First Java, 2nd Edition. O'Reilly 2005.

11. The Java tutorials (http://docs.oracle.com/javase/tutorial/)

12. Wikipedia (http://www.wikipedia.org/)

13. Wirelength and Power-Aware driven standard-cell placement, MSc Thesis, Arvanitakis Ioannis (http://www.inf.uth.gr/cced/wp-content/uploads/formidable/Arvanitakis_ioannis.pdf)

14. VEDA Laboratory (http://vedalab.inf.uth.gr/)

15. JGraphX (http://www.jgraph.com/jgraph.html)

16. ASIC World (http://www.asic-world.com/)