

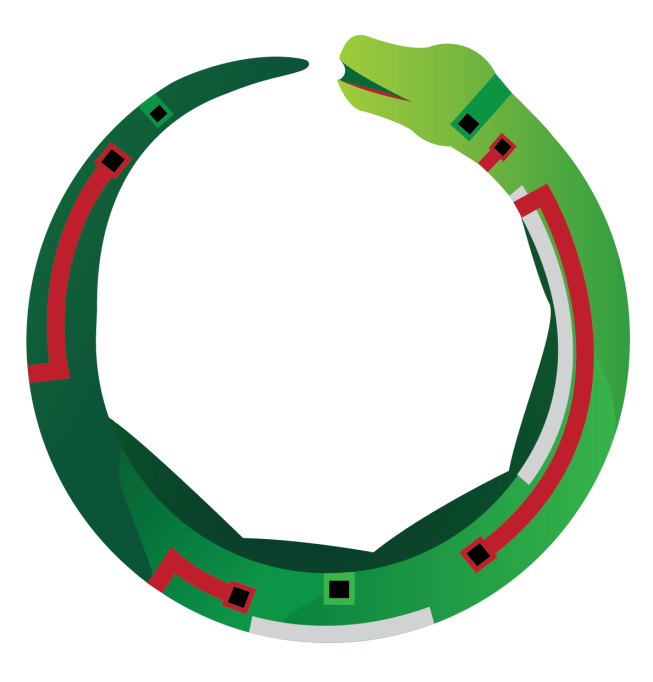
Polytech Grenoble - RICM 4: internship report

Denis Lachartre

Ophidian:

An Open-Source Library for Physical Design Research

and Teaching



2016/2017

**Acknowledgements**

I would like to express my sincere gratitude to:

Professor José Luís Güntzel,

Renan Netto,

Tiago Fontana,

Sheiny Almeida,

Michael Carboculus

Contents

[Introduction: 5](#_Toc489022683)

[I/ Presentation: 6](#_Toc489022684)

[1.1 – Federal University of Santa Catarina 6](#_Toc489022685)

[1.2 – The Laboratory 6](#_Toc489022686)

[1.3 – Ophidian 7](#_Toc489022687)

[II/ My Work: 9](#_Toc489022688)

[2.1 – The Environment 9](#_Toc489022689)

[2.2 – First Steps 10](#_Toc489022690)

[2.3 – Tests 11](#_Toc489022691)

[2.4 – The CAD Contest 11](#_Toc489022692)

[2.4.1 – The Design class 12](#_Toc489022693)

[2.4.2 – The Builder class 12](#_Toc489022694)

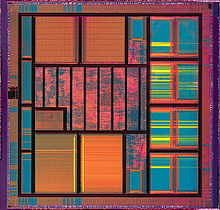
[Conclusion: 14](#_Toc489022695)

[References: 15](#_Toc489022696)

# Introduction:

For three months, I did an internship in Florianopolis, Brazil. I was in the Federal University of Santa Catarina and I worked with three Brazilian students in a project named Ophidian. It is a system that generates a physical design of an integrated circuit which is a part of a circuit creation.

Nowadays we can integrate 2e9 transistors on a single 45nm-technology chip. This number will continue to scale for the next couple of technology generations, requiring more transistors to be automatically placed on a chip and connected together. [1] Ophidian tells us how to place the components to maximize the efficiency. This is one of the most important challenges in the field of Electronic Design Automation because there is no perfect solution, it can always be improved.



*Fig 1 - A VLSI integrated-circuit*

# I/ Presentation:

## 1.1 – Federal University of Santa Catarina

The Federal University of Santa Catarina or UFSC, headquartered in Florianópolis, capital of the state of Santa Catarina, was founded on December 18, 1960, with the objective of promoting teaching, research and extension. Its community is made up of about 50 thousand people, including teachers, technical-administrative and education students. There are approximately 5,500 teachers and technicians who work in activities whose results are a reference in Brazil and abroad. [2]



*Fig 2 – Location of Florianópolis*

## 1.2 – The Laboratory

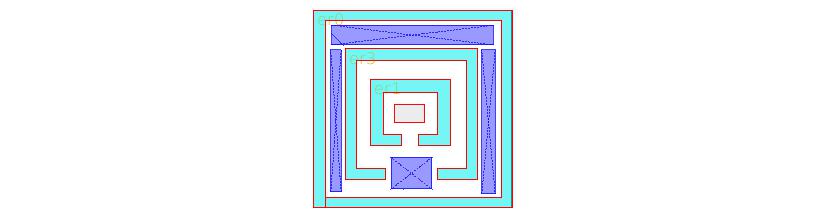
My internship took place in a laboratory in UFSC called Embedded Computing Laboratory (ECL). It is part of the Department of Informatics and Statistics (INE). They work on several topics related to integrated circuits and system (VLSI) and embedded computing such as physical synthesis Automation algorithms (gate sizing, timing analysis, and placement), hardware architectures for video coding, memory system verification and parallel algorithms.

*Fig 3 –Informatics and Statistics Department*

## 1.3 – Ophidian

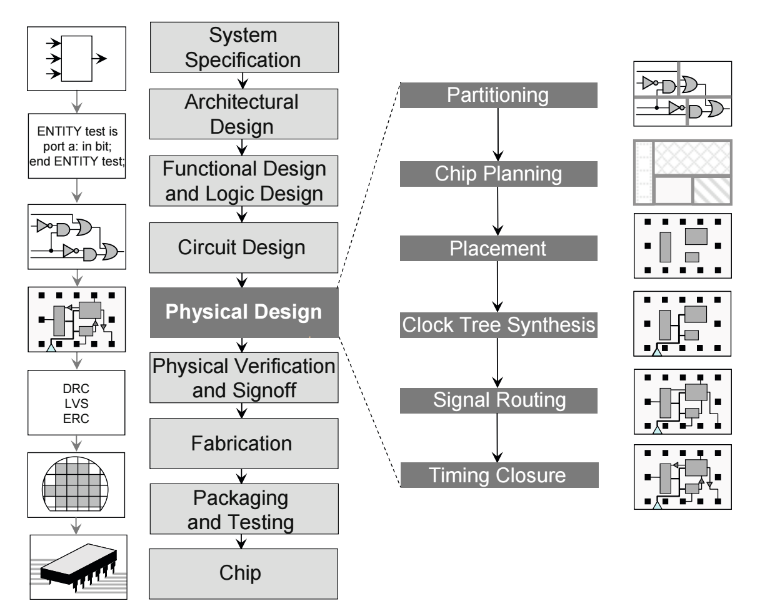
Ophidian is an open source project for physical design of integrated circuits. With this system, we can represent integrated circuits and do operations on it. Ophidian is used for both research and teaching.



*Fig 4 – A representation of a circuit*

A circuit is represented by basic blocks such as logic gates connected. There are areas where we can place them and others where we can’t. It usually is the place of other exterior blocks (memory…). The goal here is to place every block in the layout and minimize the distance between two connected blocks. Of course, it must be done in a reasonable time.

Here are the different operations needed to design a circuit. Ophidian works in the “Physical Design” part, part that requires Graph theory.



*Fig 5 – The major steps in the circuit design flow* [1]

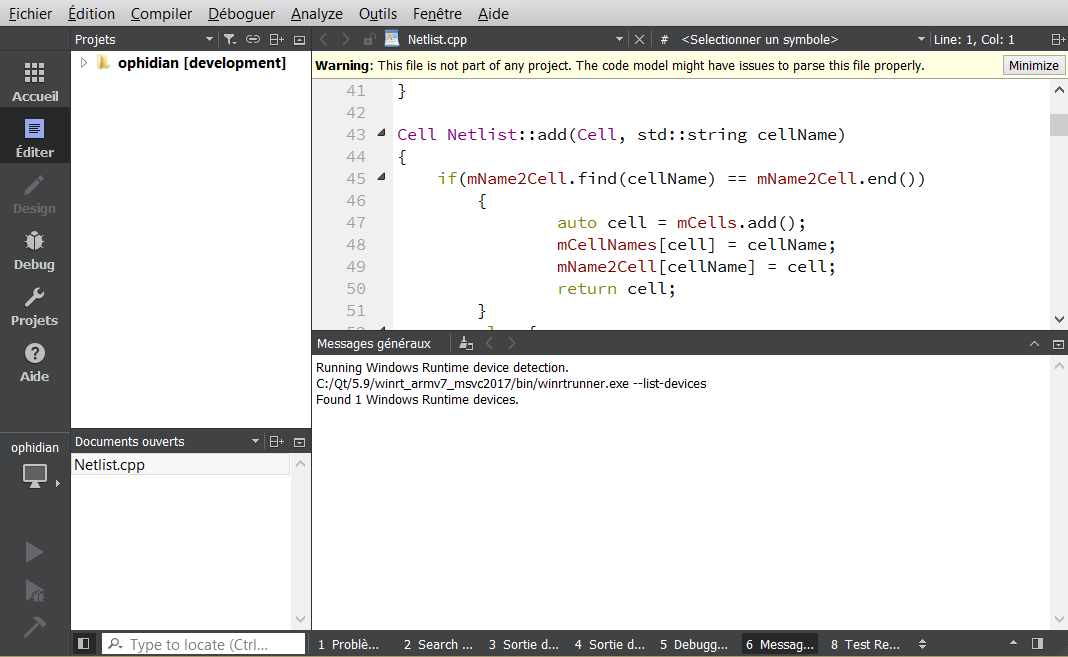
The system is divided in modules and each one has a specific use. For example the parser module will opens text files to get the data inside. The timing one provides functions for delay calculation, etc. With this, we don’t need to dive into the whole code if we want to work on it. All those modules are after compiled using CMake. It generates an executable that we can use to launch the system.

# II/ My Work:

## 2.1 – The Environment

As you may have understood, my job was to work on Ophidian. I had my own desk in the ECL laboratory. There are currently three other students in this laboratory that work in the same system. I did half of my work there and another half at my home.

The team was using an IDE called QtCreator and advised me to use it. I didn’t know about it and was really useful to code in C++, so I used it too.

****

*Fig 6 - QtCreator*

We were also using Git, a version control system (VCS) to share code and work together in the same project. The repository is managed by its owners and we had to do pull request to request changes in the final code. It protects the code from possible errors, but it’s not a 100% sure method. To guarantee that there is absolutely zero error in the final system, we were using a continuous integration service, TravisCI: Every time we do a pull request, it compiles the code in clouded machines using several compilators and block the pull request if there is an error.

During my internship, I did a total of three main pull requests.

## 2.2 – First Steps

Ophidian is a complex system and it was hard to start working on it. Indeed, I had no knowledge in C++ and integrated circuit. That is why my first two weeks was almost only reading documentations. They gave me a book to read and it was really interesting (until the Math part). It helped me understand the basics of the subject.

After that, they gave me my first task to do: Removing code style errors in the entire code, especially for the class variables. Basically, these variables were written “\_variable” and I had to change them to “mVariable”. When this task was completed, I did a pull request to merge it with the development branch in Github.

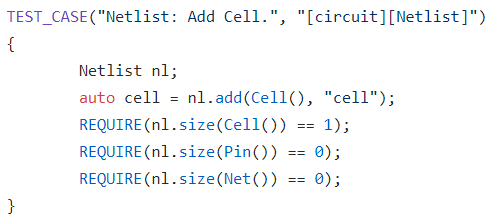
The difficulties here weren’t in the code itself but in using the environment. Finally, it taught me how to use QtCreator, how to compile Ophidian and how to do a pull request.

The second task was a bit harder. We had a problem in the code with how to represent a two dimensional point. There were too many used representations (Location, LocationMicron, LocationDbu and Point). In order to clean the code from this, I changed everything to LocationDbu (Dbu stands for DataBase Unit). To do so, I had to convert some variables because Micron and Dbu have a multiplier factor. It raised a few casting issues because it is a custom type and there are Math operations that are impossible. To compare two coordinates, for example, we first have to cast to double. Here too I did a pull request at the end.

Finally, these two tasks help me to learn how Ophidian is working because I had to work in every classes of the system.

## 2.3 – Tests

In order to know if what we do works like we want, Ophidian uses a library called Catch. Catch stands for C++ Automated Test Cases in Headers and is a multi-paradigm automated test framework for C++. In other words, it will run tests every time we compile. These tests are made by the programmer for every function that he do to validate them.



*Fig 7 – An example of test*

Here is an example; we try to add a cell to the system. If the REQUIRE lines aren’t true, the compilators will warn us that there is a problem. The strength of this library is that we know exactly what is and what isn’t working: If we change something in the A function and it breaks the B function, it will still test the B after compiling. This is the kind of error that can escape the vigilance.

It has been useful when I worked with the types. Indeed, it showed where I needed to convert or to cast the variables.

## 2.4 – The CAD Contest

The CAD contest takes place every year in the ICCAD (International Conference on Computer Aided Design). This is a multi-month, research and development competition, focusing on advanced, real-world problems in the field of Electronic Design Automation.

It first took place in 2012. Over the years, the contest has presented challenging problems and real-world benchmarks in varied topic areas while fostering productive industry-academia collaboration. Each year winners are awarded at an ICCAD special session dedicated to this contest. Thus far, it has led to more than 95 publications in top-tier conferences and journals, which have undoubtedly boosted EDA research and extended the impact of the contest. [3]



Our laboratory chose to compete this year. The contest presents three problems and we will work on “Multi-Deck Standard Cell Legalization”, which Ophidian is made for. They give us 4 files:

• Cells.lef - Physical characteristics of the technology library for the standard cell library, macros, and IO cells, etc.

• Tech.lef - Physical characteristics of the routing layers, vias, placement site types, etc.

• Design.def - Design-specific logical and physical information that represent of the design of the system.

• Design.constraints - A text file that includes some others constraints such as maximum target utilization or maximum displacement.

The system must create another def file that contains the final description of the integrated circuit.

### 2.4.1 – The Design class

Before implementing the main functions that will build our system, I needed a class that contains other existing classes (such as libraries). To do so, I created a new module in the Ophidian folder named “design” and a Design.cpp and Design.h inside of it. Here are the classes that are needed:

• **Netlist**: Defines a super-hypergraph of cells (supernodes), pins (nodes) (including pads) and nets (hyperedges).

• **Floorplan**: Provides site and rows insertion, site and rows properties and circuit dimensions.

• **Placement**: Stores placement-related properties for Netlist's entities (e.g., position, geometry).

• **Library**: Contains the geometry of the cells used in the circuit.

• **StandardCells**: Entity system that defines the Standard-Cells and its pins. [4]

In Design.h, I first created the new Design class. Inside of it, there are the constructor, the destructor and the getters. We create one object for each of the classes listed above. They are the member variables of the Design class and have their getters.

### 2.4.2 – The Builder class

# Conclusion:

Mon stage c’est une putain de blague.

# References:

[1] Andrew B. Kahng, Jens Lienig, Igor L. Markov and Jin Hu *VLSI Physical Design: From Graph Partitioning to Timing Closure,* 2011.

[2] UFSC Website : http://estrutura.ufsc.br

[3] 2017 CAD Contest Website : http://cad-contest-2017.el.cycu.edu.tw/CAD-contest-at-ICCAD2017/

[4] Ophidian’s wiki : https://github.com/eclufsc/ophidian/wiki

Q:

What is done/ not done

Images of Ophidian