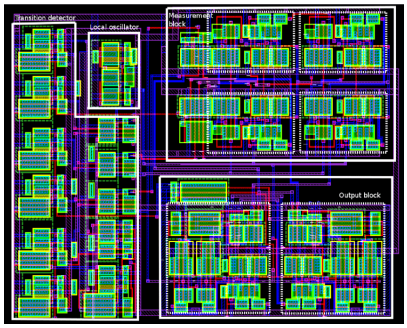
Alma Mater Studiorum · University of Bologna

VLSI (Very Large Scale Integration) design optimization problem: SMT solution



Combinatorial Decision Making and Optimization

Giuseppe Murro (giuseppe.murro@studio.unibo.it)
Giuseppe Boezio (giuseppe.boezio@studio.unibo.it)
Salvatore Pisciotta (salvatore.pisciotta2@studio.unibo.it)

August 29, 2021

Contents

1 Introduction

VLSI (Very Large Scale Integration) refers to the trend of integrating circuits into silicon chips. A typical example is the smartphone. The modern trend of shrinking transistor sizes, allowing engineers to fit more and more transistors into the same area of silicon, has pushed the integration of more and more functions of cellphone circuitry into a single silicon die (i.e. plate). This enabled the modern cellphone to mature into a powerful tool that shrank from the size of a large brick-sized unit to a device small enough to comfortably carry in a pocket or purse, with a video camera, touchscreen, and other advanced features. The aim of this project is to design a solution for the problem of put all the given input circuits in the plate optimizing its height.

2 Instances

2.1 Input

Input instances are presented using the following variables:

- w = width of the silicon plate
- n = number of circuits to insert in the plate
- x_i = horizontal dimension of i^{th} circuit
- y_i = vertical dimension of i^{th} circuit

where an instance is written in the following way:

w
n
x_0 y_0
x_1 y_1
...

2.2 Output

Starting from input instances data, once that the optimization process is finished it is given the output as:

w l
n
x_0 y_0 px_0 py_0
x_1 y_1 px_1 py_1
...

where:

- w, n, y_i, x_i are the same of the input
- l = maximum height reached by the circuits configuration
- px_i = horizontal left-down point coordinate of the i^{th} circuit block
- py_i = vertical left-down point coordinate of the i^{th} circuit block

An output instance is shown in the figure 2.1

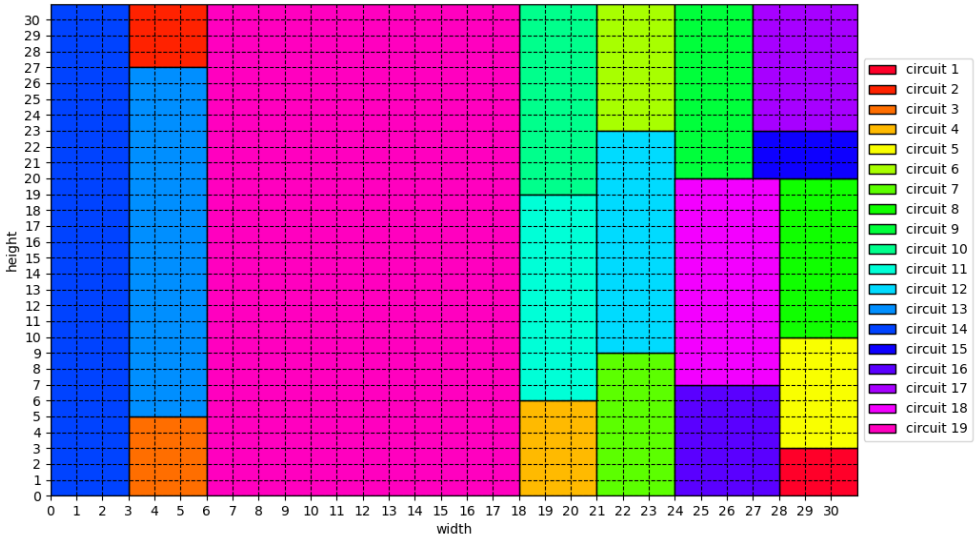


Figure 2.1: Output of an instance with w = 31 and n = 19

3 Modelling

3.1 Preliminary Reasoning

3.1.1 Modelling variables

The first part of the modelling process is the one of deciding how to model the problem and define the variables to use. In the case of SMT, variables can belong to a generic domain and they are used to express constraints in first order logic concerning logical connectives and background theories that in this case is expressed using relational operators provided by Z3Py library.

In order to describe the worst possible value of the height of the plate, it has been introduced a constant value l_{max}, computed as:

$$l_{max} = \begin{cases} max\_y & \left\lceil \frac{\sum_{i=1}^{n} y\_i}{\left\lfloor \frac{w}{max\_x} \right\rfloor} \right\rceil < max\_y \\ \left\lceil \frac{\sum_{i=1}^{n} y\_i}{\left\lfloor \frac{w}{max\_x} \right\rfloor} \right\rceil & \text{otherwise} \end{cases} \tag{3.1}$$

Where max_x and max_y are respectively the maximum value of lengths and heights of the input circuits. The choice of this value has been done taking in consideration the unluckiest case in which all circuits dimensions are max_x and max_y, so l_{max} is computed calculating the number of circuits that can be fitted horizontally and then how many rows of these blocks are needed vertically. But sometimes this approximation is not feasible if the result is less then max_y, so in that case could be used max_y as value for l_{max}.

The variables are the followings:

- **x coordinates** For each circuit its left-bottom x coordinate is searched. With n circuits:

$$\forall i \in \{1 \,..\, n\} : px_i \in \mathbb{Z}$$

- **y coordinates** For each circuit its left-bottom y coordinate is searched. With n circuits:

$$\forall i \in \{1 \,..\, n\} : py_i \in \mathbb{Z}$$

- **Height of the plate** Maximum height reached by a circuit. It is the variable to minimize

$$length \in \mathbb{Z}$$

3.1.2 Symmetries

Looking at the feasible configuration of optimal solutions it was discover that solutions with the same l value are equivalent, so there could be exponentially many equal solution and some of them are symmetries. The possible symmetries that can be detected are:

- Vertical flip
- Horizontal flip
- 180° rotation

3.2 Constraints

3.2.1 Main constraints

Using the variables that we have defined we can impose the main constraints of the problem:

- **Reduction of variables domain**: For a faster and more efficient search, aforementioned variables are constrained to get a value which is admissible with respect to the instance of the problem.
  Given w the width of the plate, x the array containing width of each circuit and y an array containing the heigth of each circuit, we have the following formula in FOL:

$$\forall i \in \{1 .. n\} : px_i >= 0 \ \wedge \ py_i >= 0$$

$$length = max(py_1 + y_1, \ py_2 + y_2, \ .., \ py_n + y_n)$$

  It can be noticed that values px_i and py_i are not upper bounded. In a first version of the solution it was imposed a limit depending respectively on the variable w and l_{max}. However, during the execution, it was noticed that time efficiency was unacceptable, so it was decided to remove them, since it was observed that without them the solution generation process was much faster.

- **Different coordinates**: In order to avoid the overlapping of the circuits in the plate it was used the predicate *Distinct* to force each pair of coordinates to be different from the others. Due to the fact that coordinates are stored in two different arrays, *Distinct* could not be used directly; for this reason the following formula has been used to combine vectors of coordinates px and py in a single vector m. The formula is the following:

$$\forall i \in \{1..n\} : m_i = mag\_w * px_i + py_i$$

  where n is the number of circuits, $mag\_w = 10^{|w_{len}|}$, w_{len} is the length of the string representing the width of the circuit and px and py are circuit coordinates.

  In this way, it has been exploited on predicate *Distinct*. The constraint is the following:

$$Distinct(m_1, m_2, ..., m_n)$$

- **Parallelism with scheduling**: concerning the optimization of the space in the plate it was thought to use something similar to the global constraint *cumulative* used in Minizinc for a CP solution.
  Looking at the task as a resource usage problem each circuit has been considered as an activity whose duration is the vertical length, its amount of resources is equals to its horizontal length while the total number of resources is w width.
  The same reason can be done in the opposite sense, so each circuit has an activity whose duration is represented by the horizontal length and amount of resources is equals to its vertical length.

The constraint has been realized in the following way using predicate provided by Z3Py:

Given S array of starting time, D array of duration of each activity, R array of resources needed for each activity and C cumulative amount of resources available in each moment, the formula is the following:

$$\forall i \in \{1..n\} : \sum_{i|S_i <= u \wedge u < S_i + D_i} R_i \ <= \ C$$

The constraints have been written as predicates:

$$cumulative(px, y, x, w)$$
$$cumulative(py, x, y, l_{max})$$

- **Overlapping**: To avoid the possible overlapping of circuits the main idea is put a relationship between the end and the beginning of each pair of circuits for both coordinates. The constraint is the following:

$$\forall i, j \in \{1..n\} \wedge i < j : px_i + x_i <= px_j \vee px_j + x_j <= px_i \vee py_i + y_i <= py_j \vee py_j + y_j <= py_i$$

3.2.2 Implied constraints

There is one important implied constraint that must be taken into account: if we draw a horizontal line and sum the horizontal sides of the traversed circuits, the sum can be at most w at each length. It can be said the same thing for the horizontal axes imposing the maximum as the l_{max}. But, if we do not consider the rotation of the blocks it is possible to not take into account this last constraint due to the construction of the l_{max} value. The implemented formulae are the followings:

$$max(px_1 + x_1,\ px_2 + x_2,\ ..,\ px_n + x_n) <= w$$
$$max(py_1 + y_1,\ py_2 + y_2,\ ..,\ py_n + y_n) <= l_{max}$$

3.2.3 Symmetry breaking constraints

In order to break the symmetries, also the ones described in 3.1.2, it was thought to put the circuit with the biggest height always on the coordinate $(0, 0)$, in this way the biggest circuit is forced to be always at the bottom left corner of the plate, rejecting all other configurations where this circuit is in another position (including 180° rotation and vertical flip of the solution).

$$px(max\_y\_index) = 0 \wedge py(max\_y\_index) = 0$$

where max_y_index is the index of the circuit with the maximum height.

To break other equivalent solutions, including the horizontal flip, it was implemented a constraint that forces the circuits whose coordinate px_i lies in the left half part of the plate to have the sum of

the covered area greater or equal than the ones in the right half side. The implemented constraint is the following:

$$\forall i \in \{1..n\} : area_i = x_i * y_i$$

$$\sum_{i=1,\, px_i <= w \, div \, 2}^{n} area_i \; >= \sum_{i=1,\, px_i > w \, div \, 2}^{n} area_i$$

3.3 Rotation model

3.3.1 Introduction

A variation of the problem consists of allowing a rotation of each circuit. This means that the final height and length of each circuit is not necessary equals to dimensions provided in input but they could be swapped.

3.3.2 Constraints

Starting from this assumption, a new array of boolean **rotation** has been added to the previous model to express the fact that original dimensions of the i^{th} circuit have been swapped or not. From a logical point of view constraints are the same of the previous model whether dimension of each circuits are not taken into account as provided in input but how they are actually used. For this reason we have introduced two new arrays storing real height and width of each circuit and they are related to the rotation array in the following way:

$$\forall i \in \{1 \,..\, n\} :$$

$$x\_r_i \; = \; if(rotation_i,\, y_i,\, x_i)$$

$$y\_r_i \; = \; if(rotation_i,\, x_i,\, y_i)$$

where x_r and y_r are the real circuit dimensions whereas x and y are dimensions provided in input.

3.3.3 Output

Output is the same of the initial model but we add to each row an R representing that the correspondent i circuit has been rotated. Following an example:

w l
n
x_0 y_0 px_0 py_0
x_1 y_1 px_1 py_1 R
...

4 Search

To compare the performance of the models, we have used the standard search method provided by Z3Py which is a class called *Optimize* which is suitable for our problem because it allows to get an assignment for each variable such that it minimizes our objective function through the method $Optimize.minimize(< objective >)$.

4.1 Results

4.1.1 Hardware used

Our experiments were conducted on a machine with those specifications:

- Intel® Core™ i7-10750H (6 core)
- RAM DDR4 16 GB
- NVIDIA® GeForce RTX™ 2060 6 GB
- Windows 10 Home 64

The version of the interpreter used is *Python 3.8.10* with *z3-solver 4.8.12.0*.

4.1.2 Execution

As it can be noticed in the image 4.1, both models solve more or less the same amount of instances within 5 minutes. The main difference is that the final model covers two more instances than the rotation one and on average it can solve these instances in less time.

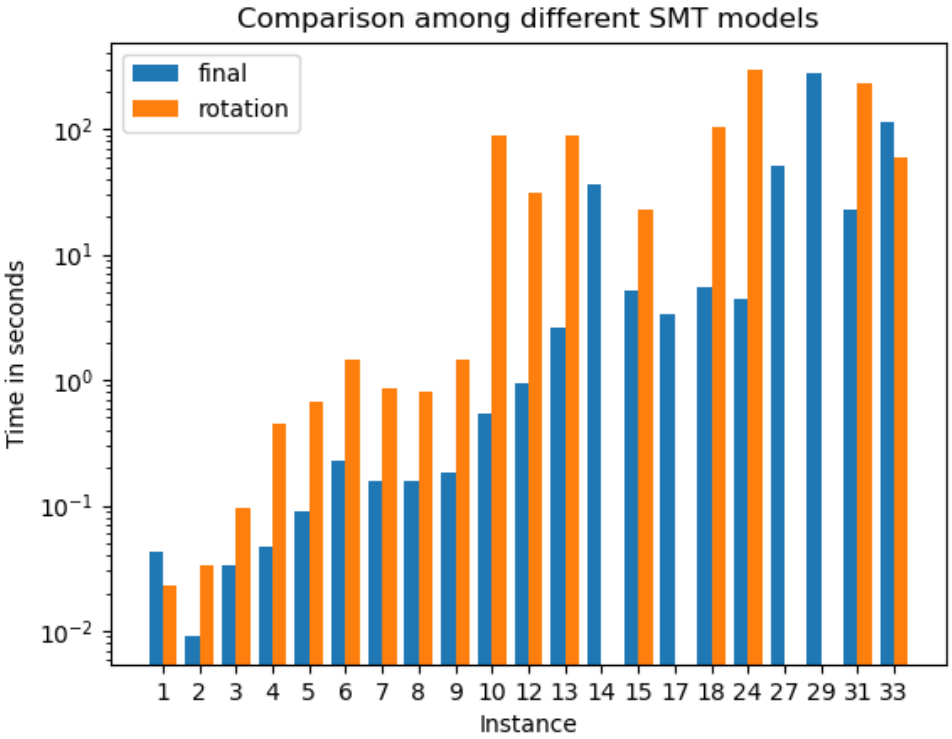


Figure 4.1: Comparison among different models