VLSI Design: a SMT approach

Davide Baldelli, davide.baldelli4@studio.unibo.it
Antonio Morelli, antonio.morelli3@studio.unibo.it
Tommaso Cortecchia, tommaso.cortecchia@studio.unibo.it
Stefano Ciapponi, stefano.ciapponi@studio.unibo.it

***Abstract*— This report describes a Combinatorial Optimization approach to the Very Large Scale Integration (VLSI) problem, exploiting Satisfiability Modulo Theories (SMT) technologies.**

***Keywords*— VLSI, SMT**

I. 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 formal problem is designed as follows: given a fixed-width plate and a list of rectangular circuits, decide how to place them on the plate so that the length of the final device is minimized (improving its portability). Consider two variants of the problem. In the first, each circuit must be placed in a fixed orientation with respect to the others. This means that, an $n \times m$ circuit cannot be positioned as an $m \times n$ circuit in the silicon plate. In the second case, the rotation is allowed, which means that an $n \times m$ circuit can be positioned either as it is or as $m \times n$.

An instance of VLSI is the width of the silicon plate w, the number of circuits n, and the horizontal and vertical dimension w_i and h_i of the i-th circuit. The solution should indicate the length of the plate l, as well as the position of each circuit by its x_i and y_i, which are the coordinates of the left-bottom corner.

The purpose of this project is to model and solve the problem using Constraint Programming (CP), propositional SATisfiability (SAT), its extension to Satisfiability Modulo Theories (SMT), and Linear Programming (LP). Solving processes that exceed a time limit of 5 minutes (300 secs) have been aborted.

II. SMT

For the SMT model, we have used the Z3 solver with LRA (Linear Real Arithmetics) logic, as backed to python. We mainly followed Banerjee et al. [1] work.

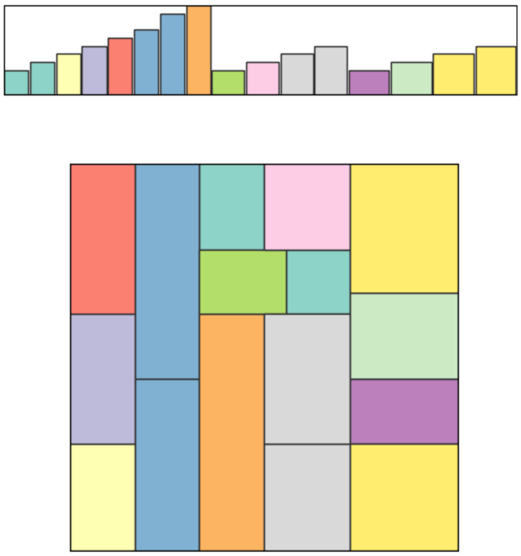

Fig. 1. Graphical representation of an instance and its solution.

A. Parameters

We receive an input in the form:

- w: width of the silicon plate;
- n: number of circuits to be placed;
- (w_i, h_i): width and height of the i-th circuit;

B. Variables

We have defined the following variables:

- h: heigth of the silicon plate, the variable to minimize;
- (x_i, y_i): coordinates of the left-bottom corner of the i-th circuit;

C. Contraints

First of all, we impose the constraints to make the circuits fit inside the silicon plate:

$$\bigwedge_{1 \le i \le n} 1 \le x_i \le w - w_i$$

$$\bigwedge_{1 \le i \le n} 1 \le y_i \le o - h_i,$$

where o is the current guess in the bisection method of the height of the silicon plate (this step will be clearer in section D.).

Then we have defined the non-overlapping constraints. As we have already seen, two circuits, the i-th and the j-th do not overlap if at least one of the following clauses holds:

$$x_i + w_i \leq x_j$$
$$x_i - w_j \geq x_j$$
$$y_i + h_i \leq y_j$$
$$y_i - h_j \geq y_j$$

In order to ensure that of the four inequalities, at least one holds true, we have firstly introduced two additional boolean variables x_{ij}, y_{ij} for each $1 \leq i \neq j \leq n$:

$$x_i + w_i \leq x_j + w(x_{ij} + y_{ij})$$
$$x_i - w_j \geq x_j - w(1 - x_{ij} + y_{ij})$$
$$y_i + h_i \leq y_j + ub(1 + x_{ij} - y_{ij})$$
$$y_i - h_j \geq y_j - ub(2 - x_{ij} - y_{ij})$$

Due to the fact that introducing $2 \cdot n \cdot (n-1)$ variables would result in an excessive additional complexity to the model, we have added in logical disjunction the conjunction of the clauses above for each possible configuration of x_{ij} and y_{ij}. More formally, if I call the conjunction of clauses above $\phi(x_{ij}, y_{ij})$, it results in:

$$\bigwedge_{1 \leq i < j \leq n} \phi(0,0) \vee \phi(1,0) \vee \phi(0,1) \vee \phi(1,1)$$

We have imposed the left-bottom corner of the largest circuit (the one with the maximum area) to lie in the bottom-left position with reference to the horizontal and vertical symmetries:

$$\hat{i} = argmax_{1 \leq i \leq n}(w_i \cdot h_i)$$
$$x_{\hat{i}} \leq (w - w_{\hat{i}})/2$$
$$y_{\hat{i}} \leq (h - h_{\hat{i}})/2$$

In this way we break the symmetries to the horizontal central axis and the vertical central axis (always but when the largest circuit is cut in half by an axis).

D. Search strategy

We have implemented the bisection method to narrow the search space for h. Starting from a tight interval revealed to be crucial for improving the performance. We have defined the lower bound of l as:

$$lb = \frac{\sum_{1 \leq i \leq n} w_i \cdot h_i}{w}$$

We can notice that this definition is mathematically equivalent to impose the empty cells in the silicon plate to be at least zero. To find a low and sound upper bound (ub) we find a naive solution to the instance and set its height as ub. The algorithm that finds the naive solution proceeds by positioning the circuits one by one in descending order by height (between circuits of equal height, we order them in descending order by width), choosing between the feasible positions the lowest, and between the lowest the left-most. Once we have defined the lower and the upper bound we can trigger the bisection method, which works as described in Algorithm 1 (with ϕ, we mean all the constraints that define the problem).

Algorithm 1 Bisection Method

```
while lb<ub do
  o := (lb + ub)/2
  result := (φ ∧ (h = o))
  if result is SAT then
    ub := o
  else
    lb := o + 1
  end if
end while
```

Algorithm 2 Naive solution

Input: n number of circuits, w width of the chip, dimensions of circuits
Output: a simple, feasible solution

```
if a circuit can't fit the width then
  return UNFEASIBLE
end if
sort(CIRCUITS)                    // by height then width
POSITIONS ← empty list
for c in CIRCUITS do
  x ← 0
  y ← 0
  appended ← false
  while appended is false do
    if existsOverlap(c,(x,y),positions) then
      if x ≥ w − c[width] then
        x ← 0
        y ← y + 1
      else
        x ← x + 1
      end if
    else
      append(POSITIONS(x, y))
      appended ← true
    end if
  end while
end for
return POSITIONS
```

E. Rotation model

To deal with the variant of the problem, we have used the usual approach: we have defined a boolean variable $flip_i$ for each circuit such that if $flip_i$ is True we swap the dimension of the i-th circuit.

F. Results

The model that doesn't allow the rotation of the circuits solves all but the 38-th and the 40-th proposed instances, while the rotation model solves 32 instances, as shown in Fig. 2.

Concerning the non-rotation model, Inconsistencies regarding the time of computation have been noticed between different runs of the program.

Sometimes the instances $37, 39$ can't be optimized within the 5-minute time constraint, other times results get computed before a 1-minute mark. Fig.2 displays some of the best runs of our model.

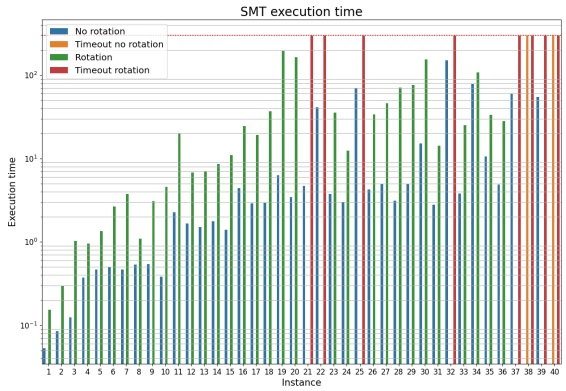

Fig. 2. Performance of the SMT model.

REFERENCES

[1] Banerjee, Suchandra, Anand Ratna, and Suchismita Roy. "Satisfiability modulo theory based methodology for floorplanning in VLSI circuits." 2016 Sixth International Symposium on Embedded Computing and System Design (ISED). IEEE, 2016.