VLSI Design: an LP 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 Linear Programming (LP) technologies.**

***Keywords*— VLSI, LP, MIP**

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. LP

To model an LP solution we have developed a Mixed Integer Programming (MIP) approach and exploited the Gurobi Optimizer, as a backend to python. The model has been extracted from Kim, Jae-Gon, and Yeong-Dae Kim [1], which proposed a meta-heuristic approach using a linear programming model embedded in simulated annealing algorithms, which are used to find a near optimal floorplan. Since this goes outside the scope of the project, we have just kept the part relative to the MIP modelling.

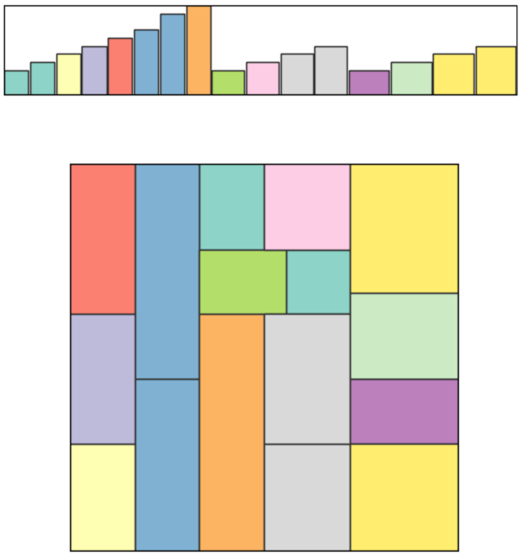

Fig. 1. Graphical representation of an instance and its solution.

A. Parameters

We receive an input in the form:

- w_c: width of the silicon plate;
- n: number of circuits to be placed;
- (p_i, q_i): width and height of the i-th circuit;

B. Variables

We have defined the following variables:

- h_c: heigth of the silicon plate, variable to minimize;
- (x_i, y_i): coordinates of the left-bottom corner of the i-th circuit;
- r_{ij}: $= 0$ if circuit i is to the left of block j, 1 otherwise;
- u_{ij}: $= 0$ if circuit i is below block j, 1 otherwise.

C. Baseline model

Minimize h_c

$$\begin{aligned}
\text{subject to } x_i + p_i &\leq w_c && \forall i && (1)\\
y_i + q_i &\leq h_c && \forall i && (2)\\
x_i + p_i - x_j &\leq w_c && \forall i,j && (3)\\
y_i + q_i - y_j &\leq h_c && \forall i,j && (4)\\
r_{ij} + r_{ji} + u_{ij} + u_{ji} &\leq 3 && \forall i,j; i \neq j && (5)\\
x_i + p_i &\leq x_j + M \cdot r_{ij} && \forall i,j,\ i \neq j && (6)\\
y_i + q_i &\leq y_j + M \cdot u_{ij} && \forall i,j; i \neq j && (7)
\end{aligned}$$

All the variables domains have been defined as the set of natural numbers, except for the binary variables r and u that are constrained to take values in $\{0, 1\}$. We have applied a simple method to reduce the domain of values of h_c. We have defined the lower bound of h_c as:

$$h_{min} = \frac{\sum_{1 \leq i \leq n} w_i \cdot h_i}{w_c}$$

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 (h_{max}) we find a naive solution to the instance and set its height as h_{max}. The algorithm that finds the naive solution proceeds by positioning the circuits one by one in descending order by height and then by width, choosing the lowest and leftmost free position. This algorithm, other than providing in a fast time a basic, feasible solution, revealed to be offering a good upper bound for the height of the chip.

With (1),(2),(3),(4) we set the boundaries for the positions of the circuits. (3) and (4) are actually redundant, but somehow helped to increase the performance of the solver, so we kept them. (6) and (7) prevent overlaps of circuits by letting each pair of elements be separated in the x or y direction. (5) ensures that at least one between $r_{ij}, r_{ji}, u_{ij}, u_{ji}$ holds. To define the non-overlapping constraints, we defined an auxiliary variable that has to be greater than every candidate h_c and w_c, M. Considering that for this problem we are already given a fixed width and we are able to compute a basic h_{max}, setting $M = 2 \cdot max(w_c, h_{max})$ is largely sufficient.

We can add a simmetry-breaking constraint which imposes the largest circuit to be placed in the bottom-left section of the chip. 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). Let $\hat{i}$ be the index of the largest block, $\hat{i} = argmax_{1 \leq i \leq n}(p_i \cdot q_i)$, then

$$x_{\hat{i}} \leq (w_c - p_{\hat{i}})/2 \quad (8)$$
$$y_{\hat{i}} \leq (h_c - q_{\hat{i}})/2 \quad (9)$$

Working on the input data it is possible to remove some variables (by fixing their value) to reduce the search space:

- with reference to the largest block constraint, we impose each circuit that it's not enough tight to stay on the left of the largest circuit, to not be placed on its left:

$$\begin{aligned}
&\texttt{if } p_i > (w_c - p_{\hat{i}})/2 \texttt{ then :}\\
&\quad r_{i\hat{i}} = 1
\end{aligned}$$

- if the sum of the length of two circuits is greater than w_c they can't be stacked together in the horizontal direction:

$$\begin{aligned}
&\texttt{if } p_i + p_j > w_c \texttt{ then :}\\
&\quad r_{ij} = 1\\
&\quad r_{ji} = 1
\end{aligned}$$

- if two circuits have equal dimensions then we can avoid symmetric assignments by reducing the choices for their relative position:

$$\begin{aligned}
&\texttt{if } p_i = p_j \wedge q_i = q_j \texttt{ then :}\\
&\quad r_{ij} = 1\\
&\quad u_{ij} = 1
\end{aligned}$$

Algorithm 1 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
```

D. Rotation model

For the rotation model, we need to add more decision variables:

- f_i: $= 1$ when the i-th circuit is rotated;
- (w_i, h_i): width and height of the i-th circuit (considering eventual rotations);

The rotation model is defined as follows:

Minimize h_c

$$
\begin{aligned}
\text{subject to } x_i + w_i &\leq w_c && \forall i && (10)\\
y_i + h_i &\leq h_c && \forall i && (11)\\
x_i + w_i - x_j &\leq w_c && \forall i,j && (12)\\
y_i + h_i - y_j &\leq h_c && \forall i,j && (13)\\
r_{ij} + r_{ji} + u_{ij} + u_{ji} &\leq 3 && \forall i,j; i \neq j && (14)\\
x_i + w_i &\leq x_j + M \cdot R_{ij} && \forall i,j,\ i \neq j && (15)\\
y_i + h_i &\leq y_j + M \cdot U_{ij} && \forall i,j; i \neq j && (16)\\
x_{\hat{i}} &\leq (w_c - w_{\hat{i}})/2 && && (17)\\
y_{\hat{i}} &\leq (h_c - h_{\hat{i}})/2 && && (18)\\
(1 - f_i) \cdot p_i + f_i \cdot q_i &= w_i && \forall i && (19)\\
f_i \cdot p_i + (1 - f_i) \cdot q_i &= h_i && \forall i && (20)
\end{aligned}
$$

Constraints (10)-(18) are the same of the baseline model with (w_i, h_i) substituted to (p_i, q_i). Constraints (19) and (20) encode the dimensions of a circuit depending whether it is rotated or not. The first two constraint reductions are no more possible, while we can always apply the rule for equal-sized blocks. Furthermore, we impose the square circuits to not rotate ($f_i = 0$), as it would be meaningless.

E. Search Strategy

We have exploited variable branching priority: the value of this attribute is used as the primary criterion for selecting a fractional variable for branching during the MIP search. For x_i, y_i, w_i, h_i, f_i we have set as priority the area of the i-th circuit, while for r_{ij} and u_{ij} the variable searching priority is the minimum between the area of the i-th circuit and the area of the j-th circuit, plus one. This way, ideally, the solver decides first the largest circuit's position, then the relative position between the two of them and finally the position of the smaller one. The priority of h_i is set to be greater than all the others.

Other than that, there is the possibility of providing an intermediate solution to the problem instance that the solver can use to make a warm start.

F. Results

The MIP model has definitely the worst performances between all the models we have tested in this work. This may be given from the fact that this is mainly a combinatorial task, and branch and bound techniques maybe are not so efficient in cutting non-optimal branches of the search space *w.r.t.* to the effort that is put to apply them. However, after exploiting the priority search, the computational results have slightly improved, with the exception of some instances in which they worsened, demonstrating, in practice, that the priority search we applied is not universally valid but simply a good heuristic.

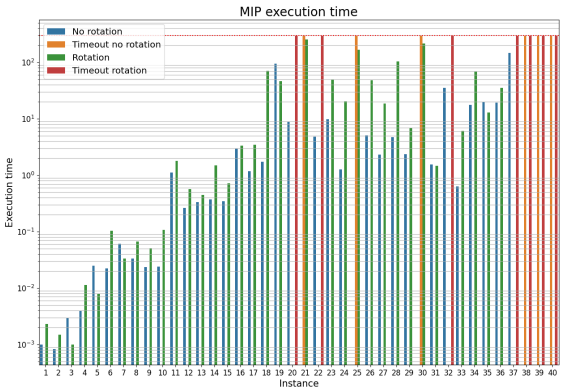

Fig. 2. Performance of the MIP model.

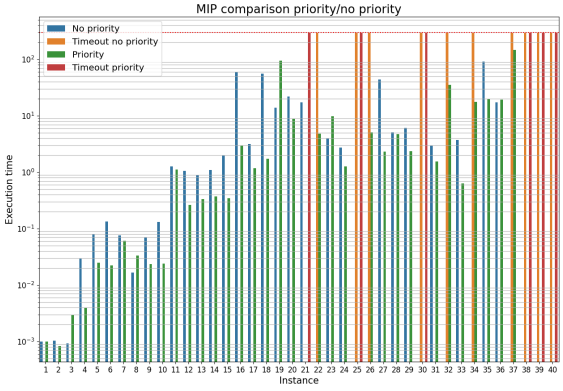

Fig. 3. Comparison between the performances of the MIP model with and without priority.

REFERENCES

[1] Kim, Jae-Gon, and Yeong-Dae Kim. "A linear programming-based algorithm for floorplanning in VLSI design." IEEE Transactions on computer-aided design of integrated circuits and systems 22.5 (2003): 584-592.