VLSI MIP

Luca Fabri, luca.fabri@studio.unibo.it
Giuseppe Morgese, giuseppe.morgese2@studio.unibo.it

September 5, 2022

1 Introduction

Linear Programming (LP) is a method to achieve the optimum of a linear objective function, subject to linear inequalities and linear equations as constraints.
A Mixed-Integer Linear Programming (MIP) problem is simply a LP problem which requires some variables to be integer.
In this report a MIP approach to solve the VLSI problem is shown.

2 Modeling

The modeling began following the advice given in the project's presentation. Thus the main variables considered are the coordinates of the bottom-left corner of each chip.

2.1 Lower and Upper bounds definition

One of the first issues addressed was deciding the variables' bounds. This step was extremely important given that it greatly reduces the search space. Regarding the chips' coordinates, the chosen ranges were:

$$x :: [0, width - min(w)] \tag{1}$$
$$y :: [0, height - min(h)] \tag{2}$$

with w and h being the array containing the chips' width and height respectively.

The bounds of the optimized variable required a more thorough work. In fact, the lower bound chosen is the maximum between the height of the tallest chip and the continuous lower bound defined as

$$L_c = \lceil \sum_j w_j h_j / W \rceil.$$

Thus the overall lower bound is

$$L_0 = max(max\_y, L_c)$$

The higher bound was computed as

$$H_0 = \left\lceil \frac{n\_circuits * max\_y}{\lfloor \frac{width}{max\_x} \rfloor} \right\rceil$$

The resulting range is:

$$h :: [L_0, H_0]$$

2.2 Basic constraints

Once the variables and their bounds were defined, the next step was to design the main constraints. The basic constraints specify that chips can't exceed the board in both directions:

$$x_i + w_i \leq width \tag{3}$$
$$y_i + h_i \leq height \tag{4}$$

Right after the no-overlap constraints were designed. To achieve such behavior, the following were implemented:

$$x_i + w_i \leq x_j \qquad or \tag{5}$$
$$x_j + w_j \leq x_i \qquad or \tag{6}$$
$$y_i + h_i \leq y_j \qquad or \tag{7}$$
$$y_j + h_j \leq y_i \tag{8}$$

2.3 Rotation

To handle rotation it had to be possible to swap w_i with h_i. To make it viable an array of boolean variables, z, was used along with the following two variables:

$$rot\_w_i = h_i * z_i + w_i(1 - z_i) \tag{9}$$
$$rot\_h_i = w_i * z_i + h_i(1 - z_i) \tag{10}$$

These variables substitute circuits' width and height in each constraint to make rotation possible.

3 Development

The MIP implementation of the model was done using PuLP. PuLP is an LP modeler written in Python that can generate LP files and call a wide choice of solvers to solve them. In this case CPLEX was chosen as a solver.

The implementation mainly followed the steps explained in section 2. A first issue rose while defining the optimized variable's lower bound. Indeed, the use of the continuous lower bound worsened the performance of the model. Thus it was not used in the MIP and the lower bound was set equal to the height of the tallest chip.

The main difference between the theoretical modeling and its implementation was the realization of the no-overlap constraints. In fact, in order to model the "or" condition a three-dimensional binary array δ multiplied by a big enough but as small as possible constants was used. To match the specific need the chosen constants were the width and the maximum height depending on the direction of the single constraint. Furthermore, the following condition had to be satisfied:

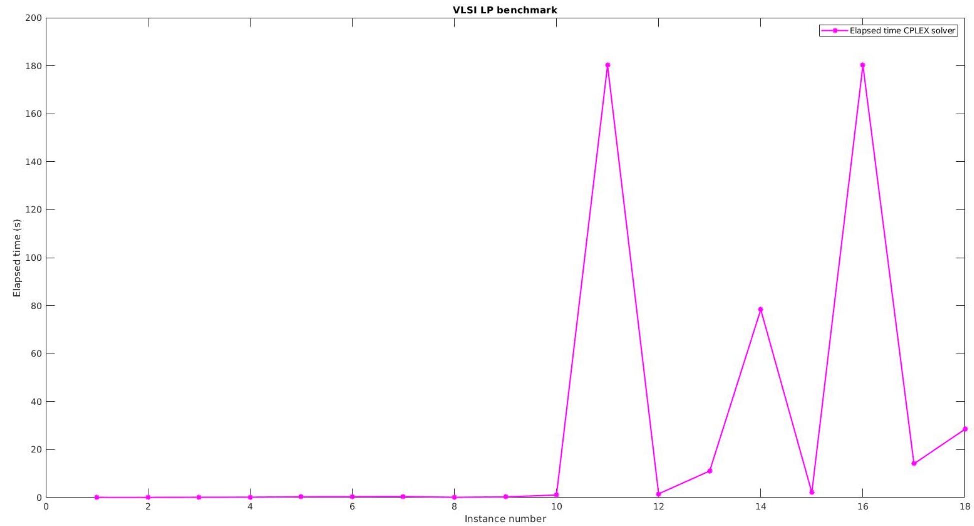
$$\sum_k \delta_i, j, k \leq 3$$

Symmetry breaking constraints were not added since they would not matter that much in a LP and worsened the performance of the solver anyway whenever tried.

Since the base MIP model couldn't solve all instances, the rotation model was not implemented. Anyway, it would not have been too different from the implementation made in the CP model.

4 Results

As shown in the graph below, the model was capable to solve 18 out of the 41 instances that were solved on a machine equipped with Intel Core i7-8550U CPU @ 1.80GHz. Most of them were solved really quickly but MIP is clearly not the best way to solve this kind of problem.



5 Conclusions

MIP is a very strong method to solve optimization problems but in this specific case is not the best given that it tries to reach an optimum without taking into account feasability thus failing in most cases.