LP Report

p.soleimanybaraijany@studio.unibo.it[1],
maryam.salehi@studio.unibo.it[2], and
shadi.farzankia@studio.unibo.it[3]

[1]University of Bologna

September 2022

1 Introduction

Very-large-scale integration(VLSI) is the process of embedding numerous transistors onto a silicon semiconductor microchip technology that emerged in the late 1970s when microcomputer chips were developing. The last few decades have witnessed remarkable growth in the electronics industry thanks to the advancements in VLSI technology. Presently, cellular communication and smartphones afford unprecedented processing capabilities and portability due to technological improvements and it is forecasted that this trend will continue as there is an ever-increasing demand for state-of-the-art devices. In VLSI design, maintaining a small area and high performance have always been two conflicting constraints considered by integrated circuit designers.

In this project, we try to propose a solution to the size problem of VLSI circuits. A certain number of circuits are embedded in a plate with a fixed width and the ultimate goal is to minimize the height of the plate.

Four different approaches including constraint programming (CP), Propositional Satisfiability (SAT), Satisfiability Modulo Theories (SMT) and linear programming (LP), have been utilized to solve the problem and the results of all approaches have been compared and analyzed to find the best solution.
This report is concerned with how Linear Programming is used to solve the VLSI problem.

2 Instance format and solution format

In this section, the format of input VLSI instances as well as the output solutions are presented.

2.1 instance format

Input instances are defined as follows:

w
n
$x_0 \quad y_0$
$x_1 \quad y_1$
...

Where:

w: width of the plate
n: number of circuits to be embedded
x_i: the horizontal dimension of the ith circuit
y_i: vertical dimension of the ith circuit

For example, a file with the following lines:

9
5
3 3
2 4
2 8
3 9
4 12

describes an instance in which the silicon plate's width is 9, and we need to place 5 circuits, with the dimensions 3×3, 2×4, 2×8, 3×9, and 4×12. Figure 1 shows the graphical representation of the instance.

2.2 Output format

After solving the problem using the input instances, the output can be represented as:
$w \quad max\_height$
n
$x_0 \quad y_0 \quad pos_{x_0} \quad pos_{y_0}$
$x_1 \quad y_1 \quad pos_{x_1} \quad pos_{y_1}$
...

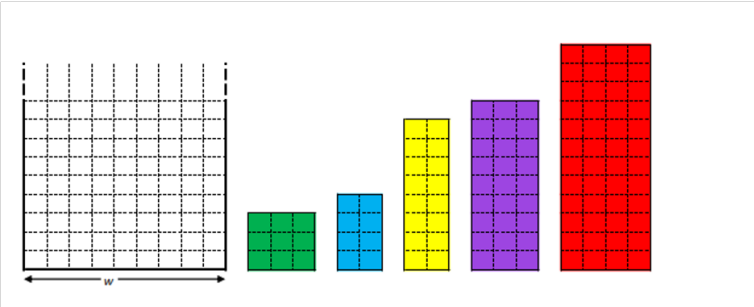

Figure 1: Graphical representation of an instance

where:

w, n, xi, yi : same as input instance
max_height: The optimal height of the plate
pos_{x_i}= horizontal coordinate of the bottom left point of the ith circuit
pos_{y_i}= vertical coordinate of the bottom left point of the ith circuit
For example, a possible solution for the instance presented in figure 1 can be as follows:

```
9    12
5
3    3    4    0
2    4    7    0
2    8    7    4
3    9    4    3
4    12    0    0
```

Which states that the maximum height is 12 and the left bottom corner of the 3×3 circuit is at (4,0). The output has been depicted in figure 2.

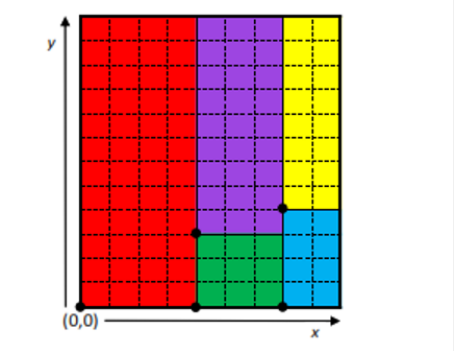

Figure 2: Graphical representation of one solution

3 Preliminary Concepts and Modeling

Operations research is different from other combinatorial decision-making and optimization approaches such as Constraint Programming (CP), Boolean Satisfiability (SAT), and Satisfiability Modulo Theories (SMT). It tries to model and solve a hard real-world optimization problem subjected to several constraints. Rather than being logic or AI-oriented, operations research is more math-oriented. It is more inequalities-centered and based on relaxation and cutting-plane rather than propagation. Linear Programming (LP) is a fundamental OR tool based on systems of linear equalities and inequalities.
An integer linear programming (ILP) problem is a type of optimization problem where the variables are integer values and the objective function and equations are linear. Integer programming is NP-complete. It is used to make processes more efficient and cost-effective. Some areas of application for linear programming include food and agriculture, engineering, transportation, manufacturing, and energy. In what follows, it is explained how Linear Programming was used to find the optimal height in the VLSI problem.

3.1 Variables and Domains

3.1.1 Maximum and Minimum height

The width of the plate is fixed and is equal to w, However, the height of the plate is a variable that should satisfy two criteria:

1. The height should be determined such that the area of the plate is minimized.

2. We should be able to embed all the circuits on the plate without overlapping.

Considering the criteria above, the plate's minimum height is the maximum of the maximum width of the circuits embedded on the plate and the sum of the areas of all the circuits divided by the width of the plate.

$$min\_height = max(y_{max}, \frac{\sum_{i=1}^{n} x_i * y_i}{w})$$

The height of the plate cannot be less than the maximum of the widths of the circuits embedded on the plate. This is because in the best case, all circuits can be placed in only one row. However, in most cases, we need to have more than one row and a good estimation of the minimum height can be achieved by dividing the sum of the areas of all the circuits by the width of the plate which is fixed.
To determine the maximum height, one needs to consider the worst case in which all of the circuits have the maximum width and only one circuit can be embedded in each row. As a result, the maximum height is equal to the sum of the maximum width of the circuits.

$$max\_height = \sum_{i=1}^{n} y_i$$

Thus, the variable height ranges from min_height to max_height.

$$min\_height \leq height \leq max\_height$$

3.1.2 Position of the Circuits

To determine where each circuit is located on the plate, only the coordinates of its left-bottom corner are kept. Variables pos_x and Pos_y denote the horizontal and vertical coordinates of the left-bottom corner of the circuits respectively.
pos_x varies between 0 and width minus min(x), where min(x) is the minimum of the lengths of the circuits.
pos_y varies between zero and maximum height minus min(y), where min(y) is the minimum of the widths of the circuits.

$$0 \leq pos_x \leq w - x_{min}$$

$$0 \leq pos_y \leq max\_height - y_{min}$$

4 Constraints

Three constraints were used to solve the VLSI problem using the linear programming approach.

4.1 The Biggest Rectangle Constraint

The biggest rectangle constraint has been used to break the symmetry. By putting the circuit with the biggest area at coordinates (0,0), vertical, horizontal, and 180 degrees symmetry can be broken. In the beginning, circuits are ordered based on their areas and the one with the biggest area will have the index 0. This circuit will be put on (0,0).

$$lp\_problem+ = pos_{x_0} == 0$$

$$lp\_problem+ = pos_{y_0} == 0$$

4.2 Boundaries Constraints

It is important to make sure that all the circuits are embedded within the plate. To satisfy this criterion two constraints were introduced to limit the coordinates of the bottom left corner of the circuits:

$$lp\_problem+ = pos_x[i] \geq 0$$

$$lp\_problem+ = pos_y[i] \geq 0$$

$$lp\_problem+ = pos_x[i] + x[i] \leq w$$

$$lp\_problem+ = pos_y[i] + y[i] \leq height$$

4.3 Non-Overlapping Constraints

Any pair of rectangular circuits should be non-overlapping. If i and j refer to two different circuits, (x_i, y_i) and (x_j, y_j) denote the widths and heights of these circuits respectively, and (pos_{x_i}, pos_{y_i}) and(pos_{x_j}, pos_{y_j}) show the coordinates of the bottom left corner of the two circuits. If circuits i and j do not overlap, at least one of the following linear inequalities should hold:

$pos_{x_i} + x_i \leq pos_{x_j}$ circuit i is to the left of circuit j
$pos_{x_i} - x_j \geq pos_{x_j}$ circuit i is to the right of circuit j
$pos_{y_i} + y_i \leq pos_{x_j}$ circuit i is below circuit j
$pos_{y_i} - y_j \geq pos_{y_j}$ circuit i is above circuit j

To ensure that at least one of these inequalities always holds, two new integer variables, a and b have been introduced that take only two values 0 and 1. There are two inequalities that must hold:

$$|pos_{x_i} - pos_{x_j}| \leq w$$
$$|pos_{y_i} - pos_{y_j}| \leq max\_height$$

Where: W is the width of the plate and max_height is equal to the sum of the widths of the circuits. Now we consider the following inequalities:

$$pos_{x_i} + x_i \leq pos_{x_j} + W * (a + b)$$
$$pos_{x_i} - x_i \leq pos_{x_j} - W * (1 - a + b)$$
$$pos_{y_i} + y_i \leq pos_{y_j} + max\_height * (1 + a - b)$$
$$pos_{y_i} - y_i \leq pos_{y_j} - max\_height * (2 - a - b)$$

As a and b can only take the values 0 or 1, we can have four different combinations of them, namely $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$. Depending on the values of a and b, only one of the above inequalities is active. For instance, if (a, b) is equal to $(0, 0)$ then only the first inequality is active and the three other constraints are satisfied. This combination allows circuit i to be anywhere to the left of circuit j. If (a, b) is equal to $(0, 1)$ then only the third inequality is active and the three other constraints are satisfied. This combination allows circuit i to be anywhere below circuit j [2].
Finding maximum height can be subject to the biggest rectangle constraint, boundaries constraint, and non-overlapping constraints.

5 Rotation

The second variant of VLSI optimization considers the case in which rotation of circuits by 90 degrees is admissible. In order to take the rotation into consideration, we introduced a new binary array roti. The solver decides whether

each element of the rotation array should be filled with zero or one. If the solver chooses zero in the rotation array for one circuit, it means the height and width of the circuit will not be swapped. If the solver chooses one in the rotation array for one circuit, it means the height and width of the circuit will be swapped. Variable rot is a Boolean array with a size equal to the number of circuits: When rotation is allowed, constraints introduced in section 4 should be revised as follows:

5.1 The Biggest Rectangle Constraint with Circuit Rotation

The biggest rectangle constraint remains unchanged when the rotation of circuits is admissible. This is due to the fact this constraint is based on the areas of circuits that do not alter when we swap the widths and heights of the rectangular circuits. Two criteria determine the necessity and possibility of rotation:

1. If the width and height of a circuit are the same, rotation is not necessary.
2. If the height of a circuit is bigger than w (width of the plate), rotation is not allowed.

5.2 Boundaries Constraints with Rotation

It is important to ensure that all the circuits are embedded within the plate. To satisfy this criterion two constraints were introduced to limit the coordinates of the bottom left corner of the circuits when rotation is admissible:

$$0 \leq pos_{x_i} + (1 - rot_i) * x_i + rot_i * y_i \leq w$$
$$0 \leq pos_{y_i} + (1 - rot_i) * y_i + rot_i * x_i \leq max\_height$$

5.3 Non-Overlapping Constraints with Rotation

Non-overlapping constraint introduced in the model without rotation can be rewritten this way:

$$pos_{x_i} + (1 - rot_i) * x_i + rot_i * y_i \leq pos_{x_j} + M * (a + b)$$
$$pos_{x_i} - (1 - rot_i) * x_i - rot_i * y_i \leq pos_{x_j} - M * (1 - a + b)$$
$$pos_{y_i} + (1 - rot_i) * y_i + rot_i * x_i \leq pos_{y_j} + M * (1 + a - b)$$
$$pos_{y_i} - (1 - rot_i) * y_i - rot_i * x_i \leq pos_{y_j} - M * (2 - a - b)$$

Where M is the maximum of the width of the plate and the maximum height:

$$M = max(W, max\_height)$$

6 Implementation and Results

PULP modeler of Python was used to solve the problem using LP approach. It offers two different solvers for CPLEX namely CPLEX_PY and CPLEX_CMD. At first, CPLEX_PY was chosen. However, as it is not free, only the first 18 instances were solved. If we wanted to solve the rest of the instances, we were required to buy the license. Consequently, we opted to download the academic edition of IBM ILOG CPLEX presented in [1] as an alternative to CPLEX_PY. CPLEX_CMD solver was used which requires to pass the path of executer. The solver was able to solve all 40 instances within the allowed time and the results were satisfactory in the model without rotation and with rotation.

References

[1] *IBM ILOG CPLEX Optimization Studio*. URL: `https://www.ibm.com/products/ilog-cplex-optimization-studio`.

[2] E. Shragowitz S. Sutanthavibul and J. B. Rosen. "An analytical approach to floorplan design and optimization". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10.6 (1991), pp. 761–769. DOI: `10.1109/43.137505`.