Combinatorial Decision Making and Optimization

Project Report

VLSI Design

Gee Jun Hui Leonidas Yunani
Anna Fabris

Academic Year: 2020 / 2021

Contents

1 Introduction

The VLSI (Very Large Scale Integration) problem refers to the trend of integrating circuits into silicon chips. Such problems involve the development of search strategies to determine the optimal circuit placements on the silicon chips. Depending on the dimensions of the circuits and the silicon chip, the search space that must be traversed may be very large. As such, the program should be optimised to determine the circuit placements within a specified time constraint, which is the main objective of the project.

Here, SAT is used to model the problem. The model will minimize the height of the silicon chip, while determining each circuit's placement on the chip.

2 Input & Output

The inputs to be fed to the program consists of a set of 40 text files. Each text file contains the following information:

- The width of the silicon chip (`width`).
- The number of circuits (`n`).
- The width and height of each individual circuit (W_c, H_c $\forall c$ in $1, .., n$).

```
8 ⟵ width
4 ⟵ n
3 3 ⟵ W_1 and H_1
3 5 ⟵ W_2 and H_2
5 3 ⟵ W_3 and H_3
5 5 ⟵ W_4 and H_4
```

The program should read each text file and output the following information:

- The width and height of the silicon chip (`width`, `height`).
- The number of circuits (`n`).
- The width, height, lower left x-position and lower left y-position of each individual circuit (W_c, H_c, Px_c, Py_c $\forall c$ in $1, .., n$).

```
8 8 ⟵ width and height
4 ⟵ n
3 3 0 0 ⟵ W_1, H_1, Px_1 and Py_1
3 5 0 3 ⟵ W_2, H_2, Px_2 and Py_2
5 3 3 0 ⟵ W_3, H_3, Px_3 and Py_3
5 5 3 3 ⟵ W_4, H_4, Px_4 and Py_4
```

Additionally, a visualisation of the solved instance is also saved as an image in the output folder.

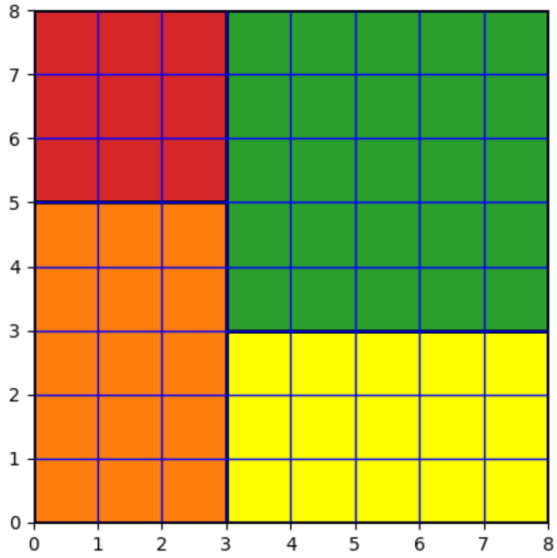

Figure 1: A visualization of a solved instance

3 Dependencies

The program has been written using the following software and libraries:

- The Matplotlib library to allow for the visualisation of the solved instances as part of the output.
- The tqdm library to allow for the tracking of the solving process for all instances.

Users can simply place the set of instances to be solved in the input folder and run the program from the command line interface.

4 SAT

4.1 Variables

Firstly, we defined some extra variable starting from the input variables:

- `circuit_widths` is an array of the *given* widths (H_c $\forall c$ in $1, .., n$) of all circuits.
- `circuit_heights` is an array of the *given* heights (H_c $\forall c$ in $1, .., n$) of all circuits.
- `max_height` is the *calculated* upper bound of the silicon chip height.

$$max\_heights = \sum_{c=1}^{n} H_c$$

- `min_height` is the *calculated* lower bound of the silicon chip height.

$$A_c = W_c * H_c$$

$$min\_height = \frac{\sum_{c=1}^{n} A_c}{max\_width}$$

Where A_c is the calculated area of the c^{th} circuit.

Following we specify that the output variable `height` ranges from `min_height` to `max_height`. To find the solution with the smallest `height` the SAT model will be used with increasing values of `height`.

The SAT model is defined as the 3D boolean array `grid[i][j][c]`. The first and second dimensions `(i, j)` represent the coordinates of the plate and range from `1` to `width` and `1` to `height` respectively, the third dimension `c` represents each circuit position and ranges from `1` to `n`.

```
grid = [[[Bool(f"grid_{i}_{j}_{c}") for c in range(n)] for j
↪  in range(height)] for i in range(width)]
```

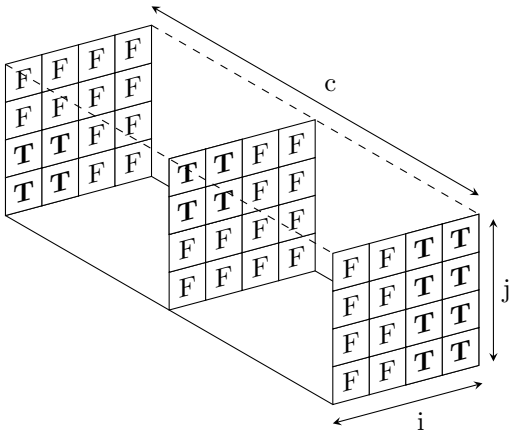

Figure 2: A representation of a possible internal state of `grid[i][j][c]` (it's possible to see where each circuit is placed by noticing the true values).

4.2 Constraints

- **At most one** circuit can occupy each place of the grid.

$$\bigwedge_{i=1}^{width} \bigwedge_{j=1}^{height} \bigwedge_{0<c1<c2\leq n} \neg(grid_{i,j,c1} \wedge grid_{i,j,c2})$$

- Every circuit must be placed in its entire form in **at least one** of its possible positions.

$$\bigwedge_{c=1}^{n} \bigvee_{i=1,j=1}^{width_{temp},height_{temp}} \overbrace{(\bigwedge_{ii=i}^{i+W_c} \bigwedge_{jj=j}^{j+H_c} grid_{c,ii,jj})}^{\text{one possible position}},$$

$$width_{temp} = width - W_c + 1,$$

$$height_{temp} = height - H_c + 1$$

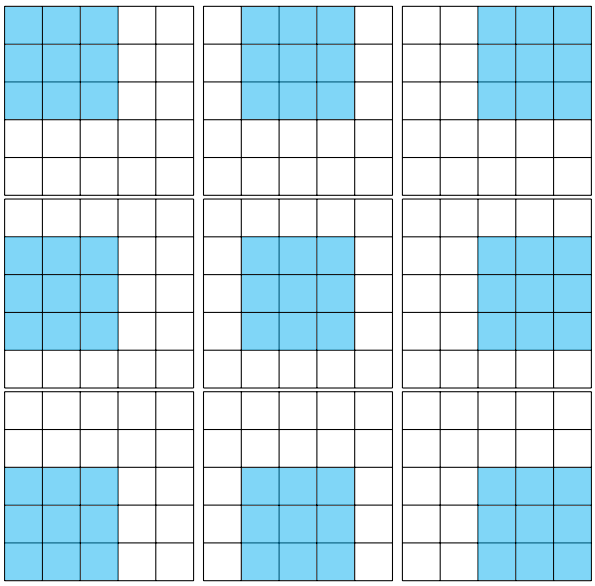

Figure 3: Example of all the possible position of one circuit in the plate.

While it may seem that **exactly one** needs to be used instead, in practice there's no need to check that maximum one circuit is in of its possible positions, because it would always lead to a worse solution.

4.3 Rotation

A simple idea to allow the model the possibility of rotation is doubling each circuit swapping its width and height coordinates and adding a constraints to avoid placing the same circuit twice.

A better solution consist in adding to the model a list of boolean variables that is the same length as the number of circuits and keeps track of whether the circuit is rotated or not.

```
rotation = [Bool(f"r_i") for i in range(n)]
```

Given this variable, it's easy to find the actual width and height of each circuit (if the circuit is rotated width and heights need to be swapped) and using $actual\_W_c$ and $actual\_H_c$ instead of W_c and H_c in the constraints.

$$\forall c \text{ in } 1..n \; actual\_W_c = \begin{cases} H_c, & \text{if rotation[c] is true} \\ W_c, & \text{otherwise} \end{cases}$$

$$\forall c \text{ in } 1..n \; actual\_H_c = \begin{cases} W_c, & \text{if rotation[c] is true} \\ H_c, & \text{otherwise} \end{cases}$$

4.4 Final Considerations

The current model was able to solve 20/40 instances with the time constraint of 300 seconds.

To improve this results another encoding of the variables was also tested: the circuits were represented with their left-bottom corners in the plate grid (as it was then used in SMT), but it had worse performance than this current model explained above.

Changing from the exactly one to the at least one constraint in the second constrain explained above helped improving the performances.

Test concerning the implied constraints seemed to generally worsen the solving time (although in some specific cases they do help), so they were removed from the code and the model.