VLSI - Very Large Scale Integration

Combinatorial Decision Making and Optimization
Module 1

Dahesh, Parsa (dahesh.parsa@studio.unibo.it)
Granata, Ludovico (ludovico.granata@studio.unibo.it)
Persiani, Simone (persiani.simone2@studio.unibo.it)

Academic year 2020-2021

Abstract

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.

1 SMT

1.1 Variables

First, let's define the problem parameters:

- *width*, the width of the board;
- *n_circuits*, defines the total number of circuits;
- *hor_dim* and *ver_dim*, the height and width of each circuit.

Now, let's define the decision variables:

- *circuitx*, x coordinate of the bottom left corner of each circuit;
- *circuity*, y coordinate of the bottom left corner of each circuit.

The goal is to minimize and find the optimal *height* of the board.

1.2 Constraints

1.2.1 Non-overlapping Constraint

The first constraint that we implemented is the non-overlapping constraint which, as the name suggests, guarantees that no circuit is overlapping inside the grid. It is developed following the MiniZinc documentation and it's implemented as follows:

$$\begin{aligned}\bigwedge_{i<j}(x_i + hor\_dim_i \leq x_j) \vee (y_i + ver\_dim_i \leq y_j) \vee \\ (x_j + hor\_dim_j \leq x_i) \vee (y_j + ver\_dim_j \leq y_i)\end{aligned} \tag{1}$$

where:

- x and y are the circuit coordinates;
- hor_dim_i and ver_dim_i are the circuit dimensions.

1.2.2 Border Constraints

The border constraints check that no circuit is partially or completely positioned outside of the grid. It's implemented as follows:

$$\begin{aligned}\bigwedge_{i}(0 \leq circuitx_i) \wedge (circuitx_i + hor\_dim_i \leq width) \wedge \\ (0 \leq circuity_i) \wedge (circuity_i + ver\_dim_i \leq height)\end{aligned} \tag{2}$$

1.2.3 Symmetry Breaking Constraint

While trying to break as many symmetries as possible in the model, we had to balance the effectiveness with the complexity of the constraints to be implemented, as complex constraints can require a combinatorial amount of time to be generated. Having evaluated many other options, we settled only on positioning the biggest circuit in the bottom-left corner of the grid, namely $(n, m) = (0, 0)$.

It is sufficient to implement the following constraints:

$$circuitx_k = 0 \land circuity_k = 0 \tag{3}$$

where:

$$k = \underset{i}{\operatorname{argmax}}\, hor\_dim_i \cdot ver\_dim_i$$

1.3 Rotation model

The rotation model introduces a new boolean decision variable called *rotated* which is *true* in case the circuit is rotated w.r.t. its starting orientation, $false$ otherwise. To correctly use the horizontal and vertical dimension variables in the previously mentioned constraints, we implemented the following constraint:

$$\begin{aligned} \bigwedge_i rotated_i \longrightarrow (hor\_dim_i = dim2_i \land ver\_dim_i = dim1_i) \land \\ \neg rotated_i \longrightarrow (hor\_dim_i = dim1_i \land ver\_dim_i = dim2_i) \end{aligned} \tag{4}$$

where $dim1$ and $dim2$ are constants read from the input instance files, that represent the non-rotated width and height of each circuit.

2 Results and Tables

Here are the tables with the final results[1] (timeout is set to 5 minutes, results are computed as the average of 3 runs):

Standard model							
Ins.	CP	SMT	SAT	Ins.	CP	SMT	SAT
1	0.295s	0.016s	0.294s	21	1.162s	timeout	timeout
2	0.297s	0.018s	0.572s	22	1.377s	timeout	timeout
3	0.300s	0.032s	2.158s	23	2.071s	8.942s	timeout
4	0.311s	0.039s	3.363s	24	0.655s	7.399s	timeout
5	0.307s	0.046s	6.707s	25	timeout	timeout	timeout
6	0.316s	0.071s	11.254s	26	2.020s	57.975s	timeout
7	0.308s	0.070s	11.510s	27	0.901s	20.394s	timeout
8	0.320s	0.100s	15.745s	28	0.956s	53.038s	timeout
9	0.317s	0.086s	23.150s	29	2.261s	65.314s	timeout
10	0.352s	0.332s	45.351s	30	timeout	timeout	timeout
11	22.743s	timeout	timeout	31	0.861s	8.249s	timeout
12	0.452s	0.688s	170.593s	32	timeout	timeout	timeout
13	0.409s	0.541s	timeout	33	0.668s	21.927s	timeout
14	0.435s	1.772s	timeout	34	timeout	timeout	timeout
15	0.456s	0.983s	timeout	35	40.135s	timeout	timeout
16	9.154s	timeout	timeout	36	43.464s	timeout	timeout
17	1.377s	5.479s	timeout	37	timeout	timeout	timeout
18	0.565s	6.516s	timeout	38	timeout	timeout	timeout
19	1.045s	timeout	timeout	39	timeout	timeout	timeout
20	1.400s	133.834s	timeout	40	timeout	timeout	timeout

[1]Run with CPU: AMD Ryzen 5 3600; GPU: NVIDIA GeForce GTX 1660 Super; RAM: 16GB; OS: Windows 10

Rotation model							
Ins.	CP	SMT	SAT	Ins.	CP	SMT	SAT
1	0.300s	0.019s	1.902s	21	timeout	timeout	timeout
2	0.311s	0.027s	3.998s	22	timeout	timeout	timeout
3	0.315s	0.047s	11.905s	23	timeout	timeout	timeout
4	0.318s	0.149s	23.888s	24	2.910s	timeout	timeout
5	0.325s	0.476s	186.306s	25	timeout	timeout	timeout
6	0.352s	0.781s	270.830s	26	timeout	timeout	timeout
7	0.446s	1.045s	timeout	27	7.723s	timeout	timeout
8	0.350s	0.826s	137.327s	28	timeout	timeout	timeout
9	18.657s	2.784s	timeout	29	timeout	timeout	timeout
10	1.774s	133.357s	timeout	30	timeout	timeout	timeout
11	timeout	timeout	timeout	31	timeout	timeout	timeout
12	2.256s	timeout	timeout	32	timeout	timeout	timeout
13	1.739s	timeout	timeout	33	timeout	timeout	timeout
14	timeout	timeout	timeout	34	timeout	timeout	timeout
15	3.192s	timeout	timeout	35	timeout	timeout	timeout
16	timeout	timeout	timeout	36	timeout	timeout	timeout
17	timeout	timeout	timeout	37	timeout	timeout	timeout
18	timeout	timeout	timeout	38	timeout	timeout	timeout
19	timeout	timeout	timeout	39	timeout	timeout	timeout
20	timeout	timeout	timeout	40	timeout	timeout	timeout