Very Large Scale Integration design in SMT

Combinatorial Decision Making and Optimization

Module 1

Alex Costanzino

alex.costanzino@studio.unibo.it

GitHub

Academic year 2020-2021

Abstract

Very Large Scale Integration (VLSI) refers to the trend of integrating more and more circuits into a single silicon chip. So far this trend has been depicted by the famous and empirical Moore's law, first devised in the 1965 [2].

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. Nevertheless, Moore's law seems to have reached its limits, due to the dissipative phenomena of semiconductors, posing another challenge, not only in the optimization of the space, but also in the optimization of the local energy on the plate [3].

In any case, space optimization enabled the modern devices 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.

Contents

1 Base model **5**

1.1 Base model with implied constraints 6

2 Symmetry-breaking constraints **8**

3 Rotation model **10**

3.1 Rotations with symmetry-breaking constraints 11

4 Final remarks and possible future developments **12**

Project work

The main task is the following: 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, ensuring an higher portability.

Setup

The project was developed on the following machine:

- Intel® Core™ i7-8565U 6 Core @ 1.80 GHz;
- NVIDIA® GeForce MX110™ 2GB;
- DDR4 8 GB RAM.

The main software used to tackle the problem is *Python 3.8.11*, and some of its basic libraries, and in particular *Z3Py*, a theorem prover from Microsoft Research [8].

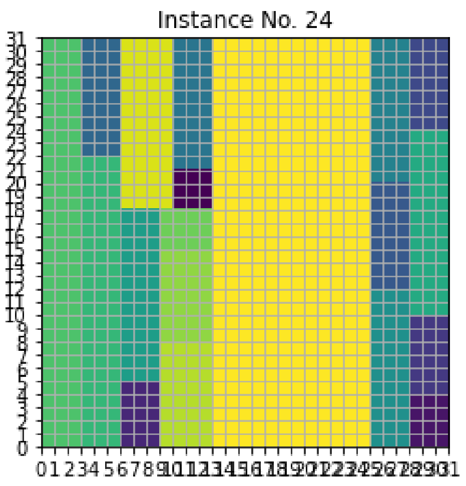
Data

Within the task, also forty instances, with increasing complexity, have been made available to test the models. The instances are under the form of **.txt* files.

Utilities

Some utilities have been developed to handle the instances and the solutions.

Solution visualizer Since the outputs are produced as text, a tool to obtain a graphical representation has also been created. The utility reads the text file, line by line, catch the data of the solution and use *Matplotlib* to visualize it.



Instance handler A set of routines to read the instances and write the solutions as **.txt* files.

Plotter A routine that plots an histogram with elapsed times for each solved instances, in log scale for better readability.

1 Base model

For the base model, the parameters fixed by each instance, are:

- Number of integrated circuits to place on the plate: n (`ICs_number`);
- Width of the plate: W (`width`);
- Widths of the integrated circuits to place: w (`IC_widths`);
- Heights of the integrated circuits to place: h (`IC_heights`).

These parameters are passed to the model through the function interface.
Other parameters, useful to limit the search space of the optimal height, are:

- An upper bound for the height of the plate: $B_{\text{up}} = \sum_{i=1}^{n} h_i$ (`upper_bound`);
- A lower bound for the height of the plate: $B_{\text{low}} = \min_{i\ldots n} h_i$ (`lower_bound`).

The decision variables are:

- The height of the plate: H (`height`);
- The horizontal positions of the integrated circuits: x (`x`);
- The vertical positions of the integrated circuits: y (`y`).

Since with *Z3Py* we cannot directly define the domain of the variables, further constraints to limit the domains are needed:

- The height H shall be bound to between the aforedefined upper and lower bounds:

$$B_{\text{low}} \leq H \wedge H \leq B_{\text{up}}$$

- The horizontal positions shall be bound between:

$$\forall i \in [1, n].\big(0 \leq x_i \wedge x_i \leq W - \min(w)\big)$$

- The vertical positions shall be bound between:

$$\forall i \in [1, n].\big(0 \leq y_i \wedge y_i \leq H - B_{\text{up}}\big)$$

The main constraints are [6]:

- All integrated circuits shall fit on the silicon plate, taking into account not only the position but also the dimensions:

$$\forall i \in [1, n].\big(x_i + w_i \le W \land y_i + h_i \le H\big)$$

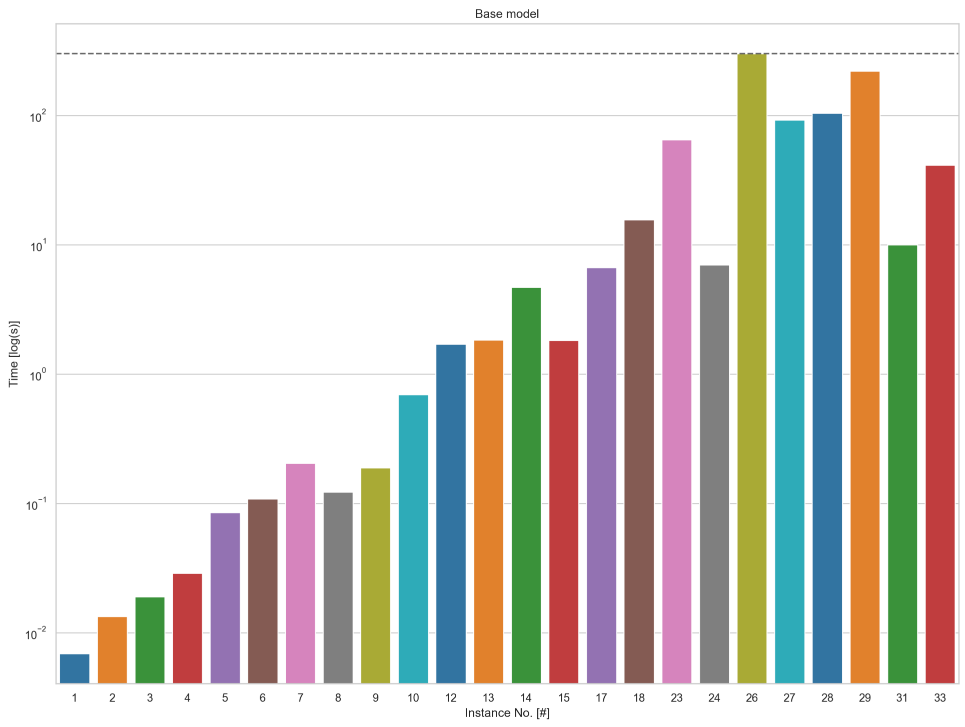
- All integrated circuits shall not overlap:

$$\forall i, j \in [1, n] : i \neq j.\big(x_i + w_i \le x_j \lor x_i - w_j \ge x_j \lor y_i + h_i \le y_j \lor y_i - h_j \ge y_j\big)$$

The goal of the problem is to place the circuits maintaining the lowest height possible for the silicon plate, therefore:

$$\min H$$

This model has been able to solve up to twenty four instances within the time boundary of 300 seconds. The performances, compared to the CP models, are much better, by number of solved instances, but also by temporal means as well.



1.1 Base model with implied constraints

It is possible to enhance the model with the following consideration: in any solution, if we draw a horizontal line and sum the horizontal sides of the traversed circuits, the sum can be at most w. A similar property holds if we draw a vertical line.

These properties can be modeled as implied constraints. These constraints are logical consequences of the initial specification of the problem [4]. Though adding an implied constraint to the specification of a constraint satisfaction problem does not change the set of solutions, it can reduce the amount of search the solver has to do.
In our case, these implied constraints can be depicted as:

$$\forall i \in [1, n].\big(\max(x_i + w_i) \leq W \land \max(y_i + h_i) \leq B_{\text{up}}\big)$$

Intuitively, the integrated circuit chosen by the max function will be the the most closer to the edge of the plate, hence the sum between its coordinate and dimension will be the sum of all sides.
Since a max function is not defined in the *Z3Py* API, an handcrafted version has been implemented. Though, in SMT modelling, these constraints have been discarded since they abruptly slowed down the solving process.

2 Symmetry-breaking constraints

A symmetry in a constraint satisfaction problem is a bijection that maps solutions to solutions and non-solutions to non-solutions. Symmetry in constraint programs can cause problems for an algorithm that searches a space of partial assignments due to redundancy in the search space [1]. One of the most popular methods for reducing symmetry is to add to the model extra constraints, so called symmetry-breaking constraints. In contrast to implied constraints, symmetry-breaking constraints are not logical consequences of the initial specification and adding them to a model may reduce the set of solutions [4].
To date, the choice of symmetry-breaking constraints has been determined solely by how much symmetry is broken, weighed against the cost of adding the symmetry-breaking constraints themselves [4]: a weaker symmetry-breaking scheme might sometimes be preferred to a stronger one because of the implied constraints that can be derived from it.
In this problem the main idea is to place the biggest component in the bottom left side of the silicon plate, and - in general - have an higher density of integrated circuits in that side of the board. Note that these constraints will likely have more effect when rotations will be allowed, since there will be more symmetries due to rotations.
From now on the model considered as baseline will be the one with global constraints and implied constraints.
To take into account symmetry-breaking constraints, another parameter is added to the model: the index of the integrated circuit with the maximum height $i^* = \arg\max_{i \ldots n} h_i$ (`max_height_index`).
The symmetry-breaking constraints are modeled as follows:

- The position of the highest component shall be in the origin:

$$x_{i^*} = 0 \wedge y_{i^*} = 0$$

  Note that this is a rather strong constraint, it could actually worsen the performances.

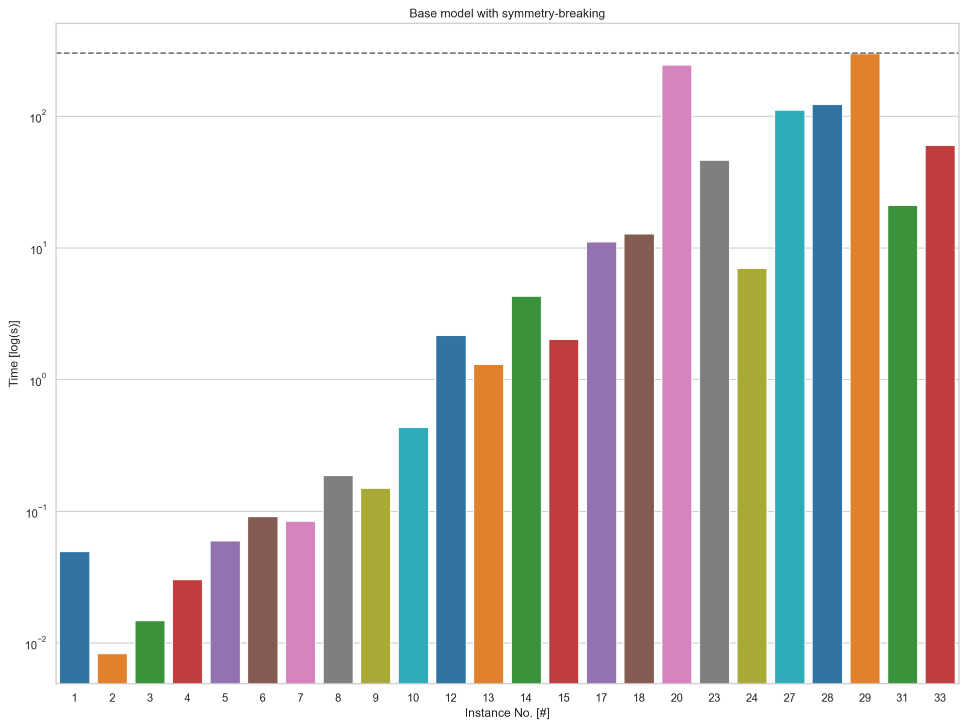
- There shall be an higher density of components on the left side of the silicon plate:

$$\forall i \in [1, n]. \left(\left(x_i \leq \frac{W}{2} \implies \sum_i w_i h_i \right) \geq \left(x_i > \frac{W}{2} \implies \sum_i w_i h_i \right) \right)$$

Since the first new constraint can be harder to optimized, it is possible to use a relaxed version of that constraint (it could also be totally discarded, but without rotations it would not improve the model so much), namely that the position of the highest component shall be in the first semi-plate [5]:

$$x_{i^*} \leq \frac{W}{2} \wedge y_{i^*} \leq \frac{H}{2}$$

In SMT the better performance was achieved with the hard symmetry-breaking constraint. The constraint enhanced some temporal performances with respect to the basic model, and slightly worsened some others.



3 Rotation model

To take into account the possible rotation of the integrated circuits, a new vector of boolean decision variables can be introduced. Each entry of the vector define the state of a piece, rotated or not rotated: r (`rotation`).
Since *Z3Py* cannot directly handle the casting with integers values in multiplication, the vector is actually implemented as a vector of integers, and then its domain has been reduced with the following constraint:

$$\forall i \in [1, n].\big(r_i = 0 \lor r_i = 1\big)$$

As far as concerns the definition of the constraints, the fitting ones change a bit:

$$\forall i \in [1, n].\big(x_i + r_i h_i + (1 - r_i) w_i \leq W \land y_i + r_i w_i + (1 - r_i) h_i \leq H\big)$$

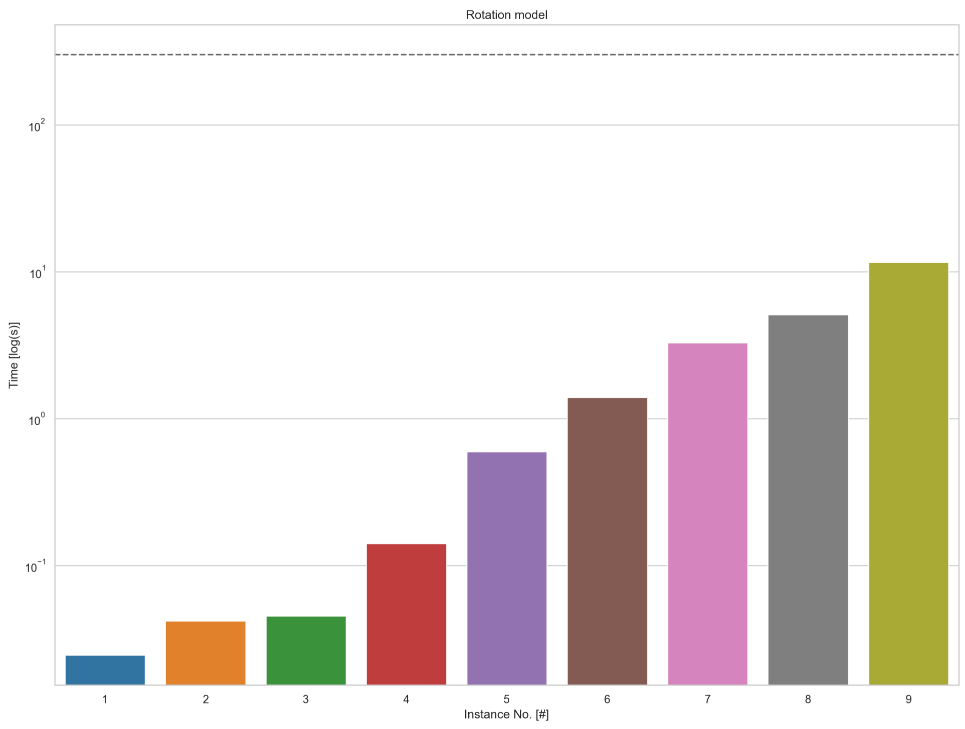
Also the implementation of the overlap constraint changes slightly:

$$\begin{gathered} \forall i, j \in [1, n] : i \neq j \\ \big(x_i + r_i h_i + (1 - r_i) w_i \leq x_j \lor x_i - r_j h_j - (1 - r_j) w_j \geq x_j \\ \lor \\ y_i + r_i w_i + (1 - r_i) h_i \leq y_j \lor y_i - r_j w_j - (1 - r_j) h_j \geq y_j\big) \end{gathered}$$

And a new implied constraint emerges, a circuit shall not rotate if its height is greater than the width of the plate, fixed by the instance:

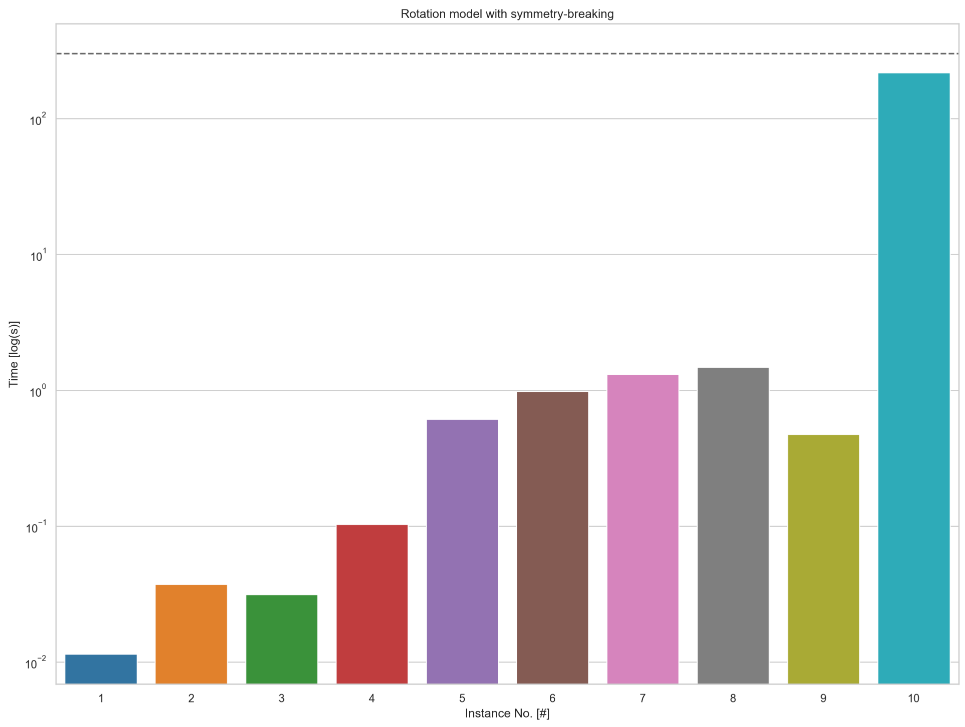
$$\forall i \in [1, n].\big(h_i > W \implies r_i = 0\big)$$

Of course, since taking into account also the rotations of the integrated circuits adds more complexity, the general performance achieved so far is worsened. Lot of instances are no longer solved and some others have drops in time performances:



3.1 Rotations with symmetry-breaking constraints

If symmetry-breaking constraints are enforced, there is a very slight improvement in some easy instances, while some others are a bit worsened, like happened in CP:



4 Final remarks and possible future developments

The best model so far has been - as expected - the base one with symmetry-breaking constraints. On its own this model has been able to solve up to 24 instances.
In general, taking into account also other deployed models, also in SMT a total of 25 instances have been solved, but with better temporal performances and an easier implementation with respect to CP modelling.
Further improvements may be achieved with:

- Use of decomposed constraints to speed-up the solver;
- Use of a dual model, where a matrix, describing the temporary placement of each integrated circuit, is instantiated. In this kind of representation more effective symmetry-breaking constraints may be enforced;
- Channeling with the aforementioned model;
- Implementation of Lodi's model [7].

References

[1] James Crawford et al. "Symmetry-Breaking Predicates for Search Problems". In: Morgan Kaufmann, 1996, pp. 148–159.

[2] R.R. Schaller. "Moore's law: past, present and future". In: *IEEE Spectrum* 34.6 (1997), pp. 52–59. DOI: 10.1109/6.591665.

[3] Kaustav Banerjee, Massoud Pedram, and Amir H. Ajami. "Analysis and Optimization of Thermal Issues in High-Performance VLSI". In: *Proceedings of the 2001 International Symposium on Physical Design*. ISPD '01. Sonoma, California, USA: Association for Computing Machinery, 2001, pp. 230–237. ISBN: 1581133472. DOI: 10.1145/369691.369779. URL: https://doi.org/10.1145/369691.369779.

[4] Alan M. Frisch, Christopher Jefferson, and Ian Miguel. "Symmetry Breaking as a Prelude to Implied Constraints: A Constraint Modelling Pattern". In: *Proceedings of the 16th European Conference on Artificial Intelligence*. ECAI'04. Valencia, Spain: IOS Press, 2004, pp. 171–175. ISBN: 9781586034528.

[5] Helmut Simonis and Barry O'Sullivan. "Using Global Constraints for Rectangle Packing". In: (Jan. 2008).

[6] Suchandra Banerjee, Anand Ratna, and Suchismita Roy. *Satisfiability Modulo Theory based Methodology for Floorplanning in VLSI Circuits*. 2017. arXiv: 1709.07241 [cs.AR].

[7] Vanessa Bezerra et al. "Models for the two-dimensional level strip packing problem – a review and a computational evaluation". In: *Journal of the Operational Research Society* 71 (Apr. 2019), pp. 1–19. DOI: 10.1080/01605682.2019.1578914.

[8] Microsoft Research. *The Z3 Theorem Prover*. 2021. URL: https://github.com/Z3Prover/z3.