VLSI Design: a SAT approach

Davide Baldelli, davide.baldelli4@studio.unibo.it
Antonio Morelli, antonio.morelli3@studio.unibo.it
Tommaso Cortecchia, tommaso.cortecchia@studio.unibo.it
Stefano Ciapponi, stefano.ciapponi@studio.unibo.it

***Abstract*— This report describes a Combinatorial Optimization approach to the Very Large Scale Integration (VLSI) problem exploiting SATisfiability (SAT) technology.**

***Keywords*— VLSI, SAT**

I. Introduction

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.

The formal problem is designed as follows: 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 (improving its portability). Consider two variants of the problem. In the first, each circuit must be placed in a fixed orientation with respect to the others. This means that, an $n \times m$ circuit cannot be positioned as an $m \times n$ circuit in the silicon plate. In the second case, the rotation is allowed, which means that an $n \times m$ circuit can be positioned either as it is or as $m \times n$.

An instance of VLSI is the width of the silicon plate w, the number of circuits n, and the horizontal and vertical dimension w_i and h_i of the i-th circuit. The solution should indicate the length of the plate l, as well as the position of each circuit by its x_i and y_i, which are the coordinates of the left-bottom corner.

The purpose of this project is to model and solve the problem using Constraint Programming (CP), propositional SATisfiability (SAT), its extension to Satisfiability Modulo Theories (SMT), and Linear Programming (LP). Solving processes that exceed a time limit of 5 minutes (300 secs) have been aborted.

II. SAT

To model the SAT solution, we have used the Z3 solver as a backend to Python, and we mainly followed Soh, Takehide, et al. [1].

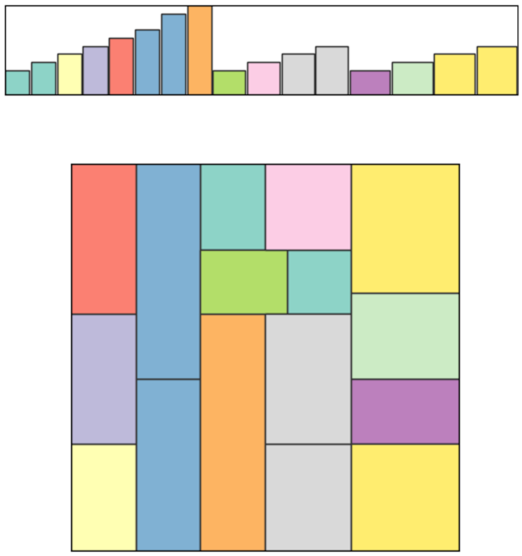

Fig. 1. Graphical representation of an instance and its solution.

A. Parameters

We receive an input in the form:

- w: width of the silicon plate;
- n: number of circuits to be placed;
- (w_i, h_i): width and height of the i-th circuit;

B. Variables

To represent the problem in propositional logic we have used the order encoding [2] to encode the coordinates (x_i, y_i) of the bottom-left corner of each circuits; this formulation uses different boolean variables $P_{x,a}$ representing $x \leq a$ for each integer variable x and each integer value a.

$$P_{x,a} \Leftrightarrow x \leq a$$

With x_i taking value in $[1, w]$ and y_i in $[1, h]$, where w is the width of the silicon plate, and h is the current guess for the height of the silicon plate (it will be clearer when we will explain how we implemented the bisection method in paragraph C).

To encode the non-overlapping constraints, we introduce two kinds of propositional variables: $lr_{i,j}$ and $ud_{i,j}$. The first one,

$lr_{i,j}$, is true if the i-th circuit is placed at the left of the j-th circuit and correspondingly, ud_{ij} is true if the i-th circuit is placed at the downward of the j-th circuit.

C. Constraints

For the sake of clarity, in the following paragraph, we are going to use the notation $A \rightarrow B$, while in our code the logical implication has been implemented through its logical definition $\neg A \vee B$.

First of all, we need the order encoding constraints, to make our encoding sound. That is to say:

$$\begin{aligned} &x_i \leq a \rightarrow x_i \leq a+1 \quad \wedge \\ &y_i \leq a \rightarrow y_i \leq a+1 \quad \forall\, 1 \leq i \leq n \end{aligned}$$

To apply those constraints to the boolean variables that we have defined above, we impose the following constraints:

$$\begin{aligned} &\bigwedge_{1\leq i\leq n} \bigwedge_{1\leq a\leq w-w_i} P_{x_i,a} \rightarrow P_{x_i,a+1} \\ &\bigwedge_{1\leq i\leq n} \bigwedge_{1\leq a\leq h-h_i} P_{y_i,a} \rightarrow P_{y_i,a+1} \end{aligned}$$

Now, to reduce the domain of x_i and y_i, as the circuits must fit inside the silicon plate, we define the following constraints:

$$x_i \leq w - w_i \wedge y_i \leq h - h_i \quad \forall\, 1 \leq i \leq n$$

To say it in propositional logic:

$$\begin{aligned} &\bigwedge_{1\leq i\leq n} \bigwedge_{w-w_i\leq a\leq w} P_{x_i,a} \\ &\bigwedge_{1\leq i\leq n} \bigwedge_{h-h_i\leq a\leq h} P_{y_i,a} \end{aligned}$$

At this point we have to design the non-overlapping constraints. For two circuits to not overlap, it is sufficient to impose that the first is placed at the right, at the left, at the downward, or at the upward of the other:

$$lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i} \quad \forall\, 1 \leq i < j \leq n$$

And to assign the right meaning to those variables we have to impose:

$$\begin{aligned} &lr_{i,j} \rightarrow x_i + w_i \leq x_j \\ &lr_{j,i} \rightarrow x_j + w_j \leq x_i \\ &ud_{i,j} \rightarrow y_i + h_i \leq y_j \\ &ud_{j,i} \rightarrow y_j + h_j \leq y_i \quad \forall\, 1 \leq i < j \leq n \end{aligned}$$

In order to translate those constrains in propositional logic, we can notice that the constraints above are equivalent to:

$$\begin{aligned} &lr_{i,j} \rightarrow (x_j \leq w_i + k \rightarrow x_i \leq k) \quad \forall\, 1 \leq k \leq w - w_i \\ &ud_{i,j} \rightarrow (y_j \leq h_i + k \rightarrow y_i \leq k) \quad \forall\, 1 \leq k \leq h - h_i \end{aligned}$$

that is to say, with our encoding:

$$\begin{aligned} &\bigwedge_{1\leq i<j\leq n} \bigwedge_{1\leq k\leq w-w_i} lr_{i,j} \rightarrow (P_{x_j,w_i+k} \rightarrow P_{x_i,k}) \\ &\bigwedge_{1\leq i<j\leq n} \bigwedge_{1\leq k\leq h-h_i} ud_{i,j} \rightarrow (P_{y_j,h_i+k} \rightarrow P_{y_i,k}) \end{aligned}$$

We have adopted three kinds of constraints to reduce the search space, choosing to mix the ones which performed better according to [1]:

- If two circuit, the i-th and the j-th, are such that the sum of their widths is greater than the width of the silicon plate, we can't pack those circuits in the horizontal direction and we can remove the constraints on $lr_{i,j}$. The same reasoning can be applied to the vertical direction.
- If two circuits, the i-th and the j-th, have the same size, we can impose $lr_{i,j}$ and omitting the constraint regarding $lr_{j,i}$, imposing the i-th circuit to be placed at the left of the j-th circuit.
- We have imposed the left-bottom corner of the largest circuit (the one with the maximum area) to lie in the bottom-left position with reference to the horizontal and vertical symmetries:

  $$\begin{aligned} &\hat{i} = argmax_{1\leq i\leq n}(w_i \cdot h_i) \\ &x_{\hat{i}} \leq (w - w_{\hat{i}})/2 \\ &y_{\hat{i}} \leq (h - h_{\hat{i}})/2 \end{aligned}$$

  In our encoding:

  $$\bigwedge_{w-w_{\hat{i}}\leq k\leq w} P_{x_{\hat{i},k}} \wedge \bigwedge_{h-h_{\hat{i}}\leq k\leq h} P_{y_{\hat{i},k}}.$$

D. Search strategy

We have implemented the bisection method to narrow the search space for h. Starting from a tight interval revealed to be crucial for improving the performance. We have defined the lower bound of h as:

$$lb = \frac{\sum_{1\leq i\leq n} w_i \cdot h_i}{w}$$

We can notice that this definition is mathematically equivalent to impose the empty cells in the silicon plate to be at least zero. To find a low and sound upper bound (ub) we find a naive solution to the instance and set its height as ub. The algorithm that finds the naive solution proceeds by positioning the circuits one by one in descending order by height (between circuits of equal height, we order them in descending order by width), choosing between the feasible positions the lowest, and between the lowest the left-most. Once we have defined the lower and the upper bound we can trigger the bisection method, which works as described in Algorithm 1 (with ϕ, we mean all the constraints that define the problem).

E. Rotation model

To deal with the variant of the problem, we proceeded as follows: we have defined a boolean variable $flip_i$ forall $1 \leq i \leq n$ such that if $flip_i$ is True, we swap the coordinates of the i-th circuit. We took in account all the possible orientations of the two components, paying attention to not exceed the size of the plate when rotating.

F. Results

After a careful research phase, the solution proposed by [1] seemed to be the most suitable for the task. The two models performed well, solving respectively 39 and 35 instances. [1] pointed out how the combinations of the three domain reduction techniques allows to obtain faster results, thinning out the number of clauses; however, the resulting models are still huge, showing some difficulties in solving the bigger instances.

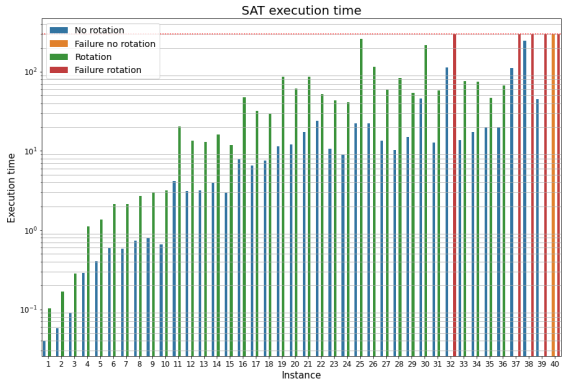

Fig. 2. Performance of the SAT model in a logarithmic scale.

Improvements are possible, for example by exploiting reusing techniques, which consists in learning lemmas called conflict clauses in order to prune redundant search space.

References

[1] Soh, Takehide, et al. "A SAT-based method for solving the two-dimensional strip packing problem." Fundamenta Informaticae 102.3-4 (2010): 467-487.

[2] Petke, Justyna, and Peter Jeavons. "The order encoding: From tractable CSP to tractable SAT." International Conference on Theory and Applications of Satisfiability Testing. Springer, Berlin, Heidelberg, 2011.

III. Appendix

Algorithm 1 Bisection Method

while lb<ub **do**
  $o := (lb + ub)/2$
  $result := (\phi \wedge (h = o))$
  if $result$ is SAT **then**
    $ub := o$
  else
    $lb := o + 1$
  end if
end while

Algorithm 2 Naive solution

Input: n number of circuits, w width of the chip, dimensions of circuits
Output: a simple, feasible solution

if a circuit can't fit the width **then**
  return $UNFEASIBLE$
end if
$sort(CIRCUITS)$ // by height then width
$x \leftarrow 0$
$y \leftarrow 0$
$POSITIONS \leftarrow empty\ list$
for c in $CIRCUITS$ **do**
  $appended \leftarrow$ **false**
  if existsOverlap(c,(x,y),positions) **then**
    if $x \geq w - c[width]$ **then**
      $x \leftarrow 0$
      $y \leftarrow y + 1$
    else
      $x \leftarrow x + 1$
    end if
  else
    $append(POSITIONS(x, y))$
    $appended \leftarrow$ **true**
  end if
end for
return $POSITIONS$