Wirelength Optimal Rectangle Packings

Julia Funke, Stefan Hougardy, Jan Schneider

Research Institute for Discrete Mathematics, University of Bonn
Lennéstr. 2, 53113 Bonn, Germany

Abstract. Finding wirelength optimal packings of rectangles is a well known problem in VLSI design. We propose a branch and bound algorithm for this problem that is based on the rectangle packing algorithm of Moffitt and Pollack. It makes use of a very efficient implementation of an incremental network simplex algorithm. Our algorithm allows for the first time to find optimum solutions of three instances of the well known MCNC block packing benchmark and optimally solves real-world instances with up to 15 rectangles in 1 hour. The largest instance so far for which an optimum solution has been computed contained 6 blocks.

1 Introduction

In this paper we consider the *fixed-outline block packing* problem of hard blocks where the objective is to minimize the total wirelength. This problem is relevant in nowadays industrial VLSI design [4] when hierarchical designs with fixed block dimensions are considered.

Our algorithm optimally solves real-world instances with 15 rectangles within 1 hour. The largest instance we can solve optimally is a real-world example with 27 rectangles, 6 of which are fixed. The largest instance that has been solved optimally so far had only 6 rectangles [7].

We will present for the first time wirelength optimum solutions to three well known block packing instances published in 1990 by the Microelectronics Center of North Carolina (MCNC) [5]. We are not only able to solve the fixed die size problems, i.e. instances with fixed width and height, but also will present optimum solutions for the much more difficult problem with variable die size where it is allowed to change the size of the placement area.

Our optimum results show that many recent algorithms behave quite poorly as they may return solutions with wirelengths more than a factor of 2 larger than the optimum. A heuristic extension of our algorithm can be used to find (not necessarily optimum) solutions for block packing instances with several hundred blocks.

2 Wirelength Optimal Fixed-Outline Block Packings

We start with a formal description of the wirelength optimization problem in fixed-outline block packing. We are given n blocks $b_1, \ldots, b_n$ of width w_i and

height h_i for $i = 1, \ldots, n$ and a rectangle R of width W and height H. A block packing assigns positions to the n blocks such that no two of them overlap and all lie within the rectangle R. More formally, if the rectangle R has as its lower left corner the point $(0, 0)$ (which we may assume without loss of generality) then we have to assign for $i = 1, \ldots, n$ a lower left corner (x_i, y_i) to block b_i such that the following conditions hold:

$$\left.\begin{aligned} x_i &\geq 0 \\ x_i + w_i &\leq W \\ y_i &\geq 0 \\ y_i + h_i &\leq H \end{aligned}\right\} \text{ for } 1 \leq i \leq n \quad \text{and} \quad \left.\begin{aligned} x_i + w_i &\leq x_j \\ \text{or } x_j + w_j &\leq x_i \\ \text{or } y_i + h_i &\leq y_j \\ \text{or } y_j + h_j &\leq y_i \end{aligned}\right\} \text{ for } 1 \leq i < j \leq n$$

The first set of inequalities guarantees that $b_1, \ldots, b_n$ lie within the rectangle R while the second set of inequalities makes sure that no two blocks overlap.

The blocks are connected by *nets* which connect a given subset of *pins*. Each pin is assigned to some block or to the rectangle R. In the latter case the pin is called *IO-pad*. The set of all pins is denoted by $\mathcal{P}$. To each pin a point (x, y) is assigned which defines the position of the pin relative to the lower left corner of the block (or of the rectangle R) it is assigned to.

The set $\mathcal{N}$ of all nets is called the *netlist*. For a net $N \in \mathcal{N}$ its *bounding box wirelength* is defined as one half of the perimeter of a smallest axis-parallel rectangle that contains all pins of the net N. The bounding box wirelength is also called *half perimeter wire length* (HPWL). Note that our problem formulation assumes that all pin shapes are points. This assumption avoids ambiguities in the definition of the wirelength.

Given a block packing its *wirelength* is defined as the sum of the bounding box wire lengths of all nets contained in $\mathcal{N}$. A *wirelength optimal block packing* is a block packing that minimizes the wirelength. The problem of finding a wirelength optimal block packing is well known to be NP-hard as it contains as a special case the NP-complete problem to decide the existence of a packing.

In [7] an exact algorithm for finding a wirelength optimal block packing has been suggested. It is a branch and bound approach with worst case runtime larger than $O(4^{\binom{n}{2}})$, where n is the number of blocks. The largest instance solved optimally with this algorithm had 6 blocks.

Our algorithm optimally solves real-world instances with up to 15 blocks within an hour. The largest real-world instance so far that was optimally solved using this algorithm had 27 blocks of which 6 had been preplaced [3]. To make this result possible one had to make use of the special structure of the instance. Figure 3 shows a wirelength optimal placement of this instance.

3 Our Approach

Moffitt and Pollack suggested a very efficient rectangle packing algorithm [6]. Their algorithm has the advantage that its runtime is independent of the sizes of the input rectangles which is important in case of the VLSI applications.

The algorithm of Moffitt and Pollack can easily be modified so that it does not only find one solution but enumerates all possibilities to pack the given rectangles into the larger rectangle. To do so one simply has to omit the clique-constraint and run the recursion until all possible relative rectangle orderings have been considered. All other pruning techniques like semantic branching and the removal of subsumed variables can still be used in our context.

The wirelength of such a relative rectangle ordering can easily be computed using the following well-known LP-formulation [1]:

$$\min \sum_{N \in \mathcal{N}} x_N^+ - x_N^- + y_N^+ - y_N^-$$

$$\text{s.t.} \quad \left.\begin{array}{l} x_N^- \le x_{b(p)} + \mathit{xoffset}(p) \\ x_N^+ \ge x_{b(p)} + \mathit{xoffset}(p) \\ y_N^- \le y_{b(p)} + \mathit{yoffset}(p) \\ y_N^+ \ge y_{b(p)} + \mathit{yoffset}(p) \end{array}\right\} \quad \text{for } N \in \mathcal{N} \text{ and } p \in N$$

$$\left.\begin{array}{l} x_i \ge 0 \\ x_i + w_i \le W \\ y_i \ge 0 \\ y_i + h_i \le H \end{array}\right\} \quad \text{for } 1 \le i \le n$$

$$b_i \ \mathit{is}\ \{left|right|above|below\}\ b_j \qquad \text{for } 1 \le i < j \le n,$$

where x_N^-, x_N^+, y_N^-, y_N^+ are variables modeling the bounding box of a net N, $b(p) \in \{R, 1, \ldots, n\}$ denotes the block to which pin p is assigned, and $\mathit{xoffset}(p)$ resp. $\mathit{yoffset}(p)$ are the offsets of p relative to the lower left corner of $b(p)$.

The dual of this LP is an uncapacitated minimum cost flow problem which can be solved in $O(n \log n(m + n \log n))$. However, the runtime of this approach is much too high to solve instances with more than 8 blocks. A significant speed up can be obtained by integrating the wirelength computation as a bounding step into the rectangle packing algorithm. Whenever the rectangle packing algorithm fixes a relation between two rectangles we solve the LP and can stop as soon as the wirelength of this solution is larger than the currently best known wirelength. This approach reduces the number of rectangle packings that have to be considered dramatically.

The LP for computing the wirelength changes only slightly between two steps of the rectangle packing algorithm. Therefore another significant speed up can be obtained by using an incremental minimum cost flow algorithm. We use the network simplex algorithm which requires exponential runtime in the worst case but turns out to be very efficient for many practical applications.

4 Experimental Results

We applied our algorithm to the three smallest instances of the well known MCNC block packing benchmark. This benchmark contains five block packing instances. The details of these instances are given in Table 1

instance	number of				die area	block area	whitespace
name	blocks	IO-pads	pins	nets			
apte	9	73	214	97	10,500 × 10,500	46,561,628	57.77%
xerox	10	2	696	203	5,831 × 6,412	19,350,296	48.25%
hp	11	45	264	83	4,928 × 4,200	8,830,584	57.34%
ami33	33	42	480	123	2,058 × 1,463	1,156,449	61.59%
ami49	49	22	931	408	7,672 × 7,840	35,445,424	41.07%

Table 1. Characteristics of the MCNC benchmark instances. Areas are in μm^2.

instance	original size	wirelength	runtime	optimal size	wirelength	runtime
apte	10,500 × 10,500	513,061	13 s	6,372 × 7,608	404,510	2 s
xerox	5,831 × 6,412	370,993	48 s	3,808 × 6,139	370,930	2 s
hp	4,928 × 4,200	153,328	102 s	4,263 × 3,108	143,302	100 s

Table 2. Optimal wirelengths for *apte*, *xerox*, and *hp* for the original die size and when rescaling of the die is allowed. Runtimes were measured on an Intel X5680 CPU at 3.33 GHz and refer to the instances with the given die sizes.

Figure 1 shows wirelength optimal packings of the three smallest instances apte, xerox, and hp, optimal wirelengths and runtimes are given in Table 2. Note that rotation or flipping of blocks is not allowed in these instances. Our results show that in [8] a wirelength for xerox is reported that is more than twice as large as the optimum wirelength. Many other, often erroneous, results for these instances have been published before.

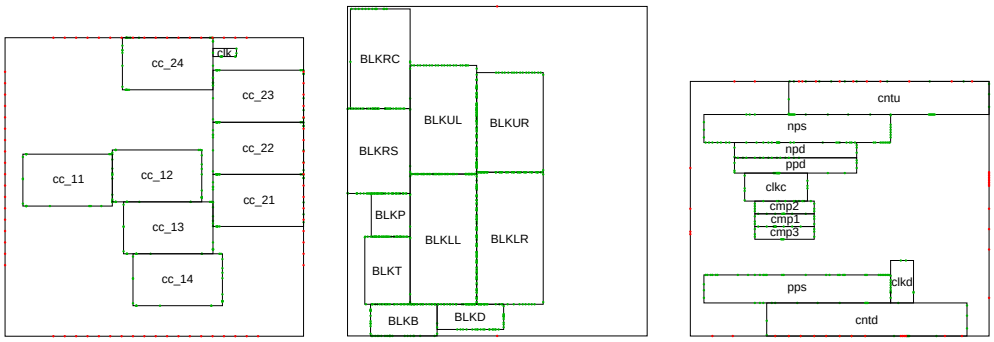
We also studied a variation of the MCNC instances where rescaling of the die size is allowed. In this case one has to rescale the positions of the IO-pads according to the resizing of the die area (see for example [2]). Let us assume that the original die area has dimensions $w \times h$ and the new die area has dimensions $w' \times h'$. Then an IO-pad with offset (x, y) to the lower left corner of the original die area is rescaled to an IO-pad with offset (x', y') to the lower left corner of the new die area, where

$$x' = x \cdot \frac{w'}{w} \quad \text{and} \quad y' = y \cdot \frac{h'}{h}. \tag{1}$$

The result is rounded to the nearest integer.

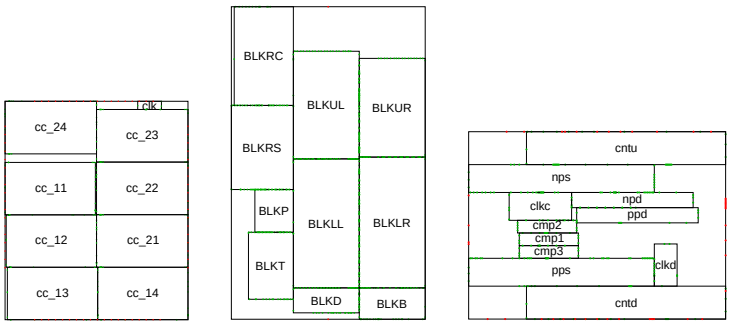
Now a straightforward approach to compute the die size that allows a wirelength optimum placement is to simply try all possible die sizes. As the width and the height of the dies is several thousand this would result in several hundred million instances that have to be solved. We therefore used an approach that makes use of the following lemma:

Lemma 1. *If l is the optimum wirelength achievable for the die size $w \times h$ for an instance with k IO-pins then the optimal wirelength for the die size $w' \times h'$ with $w' \leq w$ and $h' \leq h$ is at least $l + k(w - w' + h - h')$.*



apte *xerox* *hp*

Fig. 1. Wirelength optimal block packings of *apte*, *xerox*, and *hp* with placement area, pin locations (green dots), and IO-pad locations (red dots) as defined in the original `yal` files. Rotation or flipping of blocks is not allowed.



apte *xerox* *hp*

Fig. 2. Wirelength optimal block packings of *apte*, *xerox*, and *hp* when rescaling of the die area is allowed. IO-pad locations (red dots) are defined as described above. Rotation or flipping of blocks is not allowed.

The idea of this lemma is that if optimal wirelengths for a coarse grid of die sizes are known, e.g. when width and height are multiples of 1000, only a few cells of this grid are close enough to the best known netlength that they have to be refined by evaluating more die sizes within that cell. Using a multi-level approach based on this lemma it sufficed to compute the optimum wirelengths for only a few thousand die sizes to obtain the global optimum for each of the three instances. The optimum placements are shown in Figure 2, the die sizes and wirelengths in Table 2.

The best wirelengths that have been reported for these three instances with variable die sizes are due to Sham, Young, and Zhou (2008) [9]. They found placements with a wirelength of 483,330 for apte, 466,038 for xerox, and 186,234 for hp. This is 19% to 30% above the optimum.

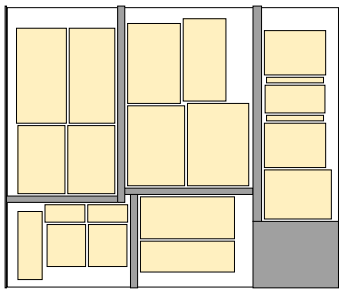


Fig. 3. An optimal placement for an instance with 27 rectangles, 6 of which are fixed (dark gray). The instance is a core macro of an actual VLSI instance provided by our industry partners at IBM. It contains 8837 pins and 3661 nets. To find an optimal solution on this instance, we had to use additional information on its structure by enumerating assignments of movable blocks to the five bins between the fixed parts.

5 Conclusion

We have presented wirelength optimal results for the three smallest instances of the well known MCNC block packing benchmark. Such results have not been known before. Our results show that recent block packing algorithms may produce packings with wirelengths more than twice as large as optimum solutions. The two largest MCNC block packing instances contain 33 respectively 49 blocks. Currently, computing wirelength optimal packings for these two instances is far beyond the realms of possibility.

6 Acknowledgments

We thank the anonymous referees for their useful comments.

References

1. A. V. Cabot, R. L. Francis, and M. A. Stary. A network flow solution to a rectilinear distance facility location problem. *AIIE Transactions*, 2(2):132–141, 1970.
2. S. Chen, Z. Xu, and T. Yoshimura. A generalized V-shaped multilevel method for large scale floorplanning. In *Quality of Electronic Design (ISQED 2009)*, 2009.
3. J. Funke, S. Hougardy, and J. Schneider. An exact algorithm for wirelength optimal placements in VLSI design. Technical report, Research Institute for Discrete Mathematics, 2011.
4. A. B. Kahng. Classical floorplanning harmful? In *ISPD*, pages 207–213, 2000.
5. K. Koźmiński. Benchmarks for layout synthesis — evolution and current status. In *28th ACM/IEEE Design Automation Conference*, pages 265–270, 1991.
6. M. Moffitt and M. Pollack. Optimal rectangle packing: a meta-csp approach. 2006.
7. H. Onodera, Y. Taniguchi, and K. Tamaru. Branch-and-bound placement for building block layout. In *DAC 1991*, pages 433–439. ACM, 1991.
8. M. Samaranayake, H. Ji, and J. Ainscough. Development of a force directed module placement tool. In *PRIME 2009*, pages 152–155, 2009.
9. C.-W. Sham, E. F. Y. Young, and H. Zhou. Optimizing wirelength and routability by searching alternative packings in floorplanning. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 13(1), 2008. Article 21.