Implementation of a Transistor Placement Method Based on Boolean Satisfiability into ASTRAN CAD Tool

Andrei A. O. Bubolz, Gustavo H. Smaniotto, Leomar S. da Rosa Jr., Maicon S. Cardoso, Felipe de S. Marques
Group of Architectures and Integrated Circuits (GACI)
Federal University of Pelotas (UFPel)
Pelotas, Brazil
{aaobubolz, ghsmaniotto, leomarjr, mscardoso, felipem}@inf.ufpel.edu.br

***Abstract*—This paper presents an alternative transistor placement method for the open source automatic layout generation tool ASTRAN, which currently implements a Threshold Accepting methodology for this purpose. Although it reaches an optimized solution, ASTRAN does not guarantee the minimum-width solution. The method presented in this paper is based on Boolean Satisfiability and ensures the minimum-width multi-row transistor placement by formulating the problem into a set of variables and clauses described through four design constraints. Experiments comparing the proposed method and the current ASTRAN placement have shown average gains of 3.02% and 12.05% in the cell area and contact number, respectively. Furthermore, it has presented a slightly overhead of 0.82% on average concerning the wirelength.**

***Index Terms*—Transistor Placement, Boolean Satisfiability, SAT, EDA Tool, ASTRAN.**

I. Introduction

Optimizations in the logic and physical synthesis are crucial factors for improving VLSI circuits. Recently, several papers pointed out that the on-the-fly design of complex gates provides improvements in different aspects of the digital design such as area, power, and delay.

One of the most important methods for the optimized physical synthesis of complex gate cells is the transistor placement [1]. Through this procedure, it is possible to obtain gains in different attributes of the final layout (wirelength and number of contacts).

The first placement method was proposed by Uehara and vanCleemput [2], where the width minimization of dual networks is obtained through a graph-based approach that aims to find a solution with the minimum number of diffusion breaks. It also proposes the linear-matrix (1D) layout style, where the transistors are placed exclusively in one direction along two P/N diffusion rows.

Logic gates in real circuits are sized to optimize circuit performance [3] and hence, transistor sizing and folding must be efficiently handled. In this scenario, Gupta and Hayes [4] proposed a method to generate layouts without any structure constraints, producing cells with a different number of P and N-type transistors and divergent transistor widths (via folding).

Regarding the width minimization for dual circuits, Carlson [5] addresses this problem for non-series-parallel planar circuits through an heuristic-based algorithm. Although obtaining optimized results, most of these methodologies do not guarantee the minimum width placement of the transistors, i.e. the optimal solution in terms of area. Iizuka, Ikeda and Asada [6], [7] proposes a Boolean Satisfiability (SAT) approach introducing an optimal placement method for both dual and non-dual transistor networks. This technique consists in to describe a specific problem into a set of Boolean expressions (clauses) and determine whether there are variable assignments that satisfy all these clauses.

Several solutions were proposed integrating placement methods with a complete cell design flow [8]–[10]. These CAD tools uses different techniques for placement in order to produce optimized layouts for dual and non-dual logic networks. For instance, while Layout Synthesizer [8] heuristically pair transistors to be placed, LiB [9] identifies strongly connected clusters and form pairs of each cluster. ASTRAN [10], a CAD tool for complex gate design, uses a Threshold Accepting technique to determine a placement solution that maximizes the diffusion sharing and minimizes the interconnection length. This methodology is based on a local search method that starts from a random feasible placement and then explores its neighborhood looking for a better solution. However, this method eventually gets stuck in a local minimum, not reaching the optimal placement.

The transistor placement methodology proposed in this paper aims to avoid this problem using a Boolean Satisfiability approach originally introduced by Iizuka, Ikeda and Asada [6], [7]. Besides that, our proposition is able to deal with CMOS transistor networks with any structure, ensuring the minimum number of diffusion breaks and the optimal solution in terms of area in a feasible computational time.

II. Proposed Methodology

This section will describe all the steps of our placement methodology. It is important to point out that our approach is based on the method described in [6], [7]. It was proposed a simplification of its constraints in order to optimize the methodology (in terms of number of variables and clauses) for dual and non-dual 1D layouts.

A. Problem Definition

The input of the proposed method is a transistor-level description of the complex gate, i.e. a spice netlist containing the pull-up and pull-down transistors.

In the 1D layout style adopted (Fig. 1), the transistors are divided into two rows containing the PMOS and the NMOS devices, respectively. Each transistor is expressed by a set of main variables that represent all possible placement configurations, as shown in Table I. The initial size of this set depends on the number of transistors of the netlist.

To generate the set of SAT clauses, four steps are performed. The first one is a simplification step, while the remaining implements the placement constraints. In this process, internal variables are created in order to represent intermediate results of these clauses, without meaning to the placement problem.

B. Variable Transistor

This component operates like a gap in the placement problem: diverging from the "conventional" transistor, it might assume dynamic values in their terminals, allowing to be placed in any position of the cell, i.e. the neighborhood of any other transistor. Moreover, the flipped variables of these components are not created, avoiding unnecessary and redundant clauses. There are two distinct situations in which a variable transistor may be employed: (1) in case of there are different number of P-type and N-type transistors in the given input, where the necessary number of components is added to complete the equality; (2) after each iteration resulting in an unsatisfiable placement, in which a variable transistor is added in each row, increasing the number of columns of the cell and consequently, the width of the potential solution. After this, the algorithm redefines variables and clauses in order to proceed with the computation.

C. Variables Reduction

This step is executed only in the first iteration, where at least one row is free of gaps. The goal of this procedure is the reduction of the total number of variables, which is performed through the analysis of each position where a transistor can be placed. First, the source and drain identifiers of all the transistors in the same row are stored. Then, for the terminals that appear only once, defined as unique terminals, two possible scenarios (according to the size of this list) are taken.

Zero unique terminals: there is nothing to be simplified. The algorithm proceeds to the creation of the placement constraints.

TABLE I
SET OF VARIABLES USED FOR THE FORMULATION

Name	The condition that assigns the variable true	Set size
$T_p(i, c, f)$	PMOS i placed in column c with orientation f	$2P^2$
$T_n(j, c, f)$	NMOS j placed in column c with orientation f	$2N^2$

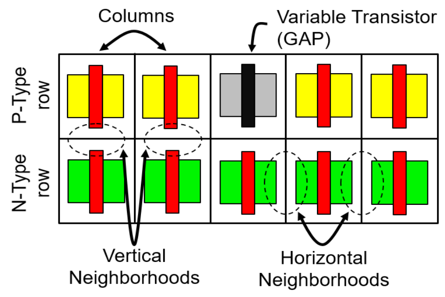

Fig. 1. 1D layout style adopted.

One or two unique terminals: transistors with unique terminals can be placed exclusively at the extremities (first and last column).

To exemplify this, consider the PMOS transistor network illustrated in Fig. 2, where we can see that at least one of its internal nodes (*p1* and *p2*) are present in all switches. On another hand, the external nodes (*VDD* and *output*) appear only once, i.e. they are considered unique terminals of the network. In this scenario, through the procedure described, the number of main variables decrease from 100 to 72, while the number of clauses reduces from 3797 to 2525, corresponding to a 28.0% and 33.5% reduction, respectively.

Therefore, a considerable amount of variables can be eliminated since the fact that these transistors cannot be placed in the intermediate positions of the cell.

D. Transistor Allocation Constraint

Each transistor must be allocated exclusively in one column. The clauses that implement this constraint are presented below (1).

$$\bigwedge_{i=1}^{P}\left\{\bigwedge_{f\in(f,nf)}\left\{\bigvee_{c=0}^{P-1}T_p(i,c,f)\wedge\right.\right.$$
$$\left.\left.\left\{\bigwedge_{j=0}^{P-2}\bigwedge_{k=j+1}^{P-1}\left[\neg T_p(i,j,f)\vee\neg T_p(i,k,f)\right]\right\}\right\}\right\} \tag{1}$$

The same formula is applied to all N-type transistors.

E. Empty Column Constraint

Columns cannot have overlapping transistors. The clauses that implement this constraint are presented below (2).

$$\bigwedge_{c=1}^{P}\left\{\bigwedge_{f\in(f,nf)}\left[\bigvee_{i=1}^{P}T_p(i,c,f)\right]\right\} \tag{2}$$

The same formula is applied to all N-type columns.

F. Horizontal Neighborhood Constraint

All transistors must have its lateral neighbors with equivalent source and drain values, with the exception of those placed in the extremities, which only need one of these.

First, all the m pairs of placement configurations are arranged in a list according to their source and drain values.

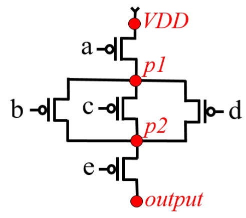

Fig. 2. PMOS logic plan of the complex gate implementing $f = !(a + (b.c.d) + e)$.

Considering all the possible associations, its necessary to place exclusively one of these in each neighborhood. The clauses that implement this constraint is presented below (3).

$$\bigwedge_{N=1}^{P-1}\left\{\bigwedge_{f\in(f,nf)}\left\{\bigvee_{i=0}^{m-1}U_i(T_x,T_y,N)\bigwedge\right.\right. \\ \left.\left.\left\{\bigwedge_{j=0}^{m-2}\bigwedge_{k=j+1}^{m-1}\left[\neg U_i(T_x,T_y,N)\vee\neg U_i(T_x,T_y,N)\right]\right\}\right\}\right\} \tag{3}$$

where $U_i(T_x, T_y, N)$ represents one of the m possible associations between the transistors $T_p(x, c, f)$ and $T_p(y, c+1, f)$ in the neighborhood N (between c and $c+1$) and P is the current number of P-type transistors.

The same formula is applied to all N-type neighborhoods.

G. Vertical Neighborhood Constraint

All P and N-type transistors placed in the same column must share their gate terminals. In other words, if there is a transistor T_{px} placed in the column C, a transistor T_{ny} with the same gate must be placed in the column C.

First, all m pair of possibilities are arranged in a list according to their gate values. Considering all the possible associations, its necessary to place exclusively one of these in each column. The clauses that implement this constraint are presented below (4).

$$\bigwedge_{C=1}^{P}\left\{\bigwedge_{f\in(f,nf)}\left\{\bigvee_{i=0}^{m-1}U_i(T_{px},T_{ny},C)\bigwedge\right.\right. \\ \left.\left.\left\{\bigwedge_{j=0}^{m-2}\bigwedge_{k=j+1}^{m-1}\left[\neg U_i(T_{px},T_{ny},C)\vee\neg U_i(T_{px},T_{ny},C)\right]\right\}\right\}\right\} \tag{4}$$

where $U_i(T_{px}, T_{ny}, C)$ represents one of the m possible associations between the transistors $T_p(x, c, f)$ and $T_n(y, c, f)$ in the column C and P is the current number of P-type transistors.

H. Satisfiability Verification

After computing all the steps presented above, the CNF file containing all the clauses is written in order to be computed by the SAT solver. The solver returns the first satisfiable assignments for the variables if it exists. In case of unsatisfiability, variable transistors are applied according to the procedure described in subsection 3.B.

The complete pseudocode of our method is shown in Algorithm 1.

III. Experimental Results

Layout analysis allowed us to evaluate the proposed solution, comparing it to placement methodology implemented into the ASTRAN tool. Through this procedure, it was possible to characterize important geometrical parameters: total cell area, total wirelength, and the number of contacts.

In order to obtain the placement solutions of our methodology, we have used CryptoMiniSat [11] as SAT solver. This tool can efficiently handle large quantities of clauses through Gaussian elimination techniques.

The experiments were performed under a well-known benchmark containing 53 handcrafted complex gate cells [12]. The transistor sizing was performed through the Logical Effort methodology [13]. Folding procedure was not applied for any of the cases. We adopted the STMicroelectronics 65nm technology node for the cell synthesis. Fig. 3 shows the obtained results which represent the optimization percentage regarding the layout obtained through the proposed methodology relative to the solutions produced by the placement method of ASTRAN. The improvements achieved by our proposition are illustrated by the positive y-axis values (blue columns), while the overheads are represented by the negative y-axis values (red columns). It is important to notice that the last column represents the average values of each analyzed parameter.

The first observed aspect was the layout area, where our approach delivers a cell with 3.02% less area (on average) compared to the original method, with a standard deviation of 5.67%. In most of the cases (88.68%) an equal or smaller area was obtained. On another hand, in some solutions, despite getting an improvement in terms of the number of diffusion breaks, a larger area was obtained. This was caused due to the fact that the proposed algorithm do not consider the cell routing and compaction, routines that potentially increases the layout wirelength, leading to increments in the total layout area. The best-case scenario has presented an optimization of 16.20%, while the worst case showed an increase of 8.10% in this aspect.

Algorithm 1 Pseudocode of the Placement Process

```
1:  $transistors \leftarrow$ getTransistorList( SpiceFile )
2:  addVariableTransistors( getListWithFewerTrans() )
3:  $SAT \leftarrow$ false
4:  while ( !$SAT$ ) do
5:      $V \leftarrow$ createVariableList( $transistors$ )
6:      $C \leftarrow$ createClauseList( $V$ )
7:      $result \leftarrow$ executeSATSolver( $V$, $C$ )
8:      if ( isSatisfiable( $result$ ) ) then
9:          $SAT \leftarrow$ true
10:     else
11:         addVariableTransistors( $transistors$ )
12:     end if
13: end while
14: $finalPlacement \leftarrow$ computeResult( $transistors$, $result$ )
```

The second observed aspect was the total wirelength (considering metal 1 and polysilicon), where our methodology obtained an overhead of 0.82% (on average) in comparison with the solutions produced by the original method, with a standard deviation of 7.02%.

Finally, the last observed parameter was the total number of contacts, where we obtained a reduction of 12.05% on average, with a standard deviation of 9.05%.

IV. Conclusions and Future Works

This paper proposes an alternative transistor placement method for the ASTRAN cell generator aiming geometrical improvements in the layouts produced by this tool. ASTRAN currently implements a Threshold Accepting technique to compute its placement solutions. However, in some cases it is possible to get stuck in a local minimum solution, and consequently, not finding the optimal placement. To avoid this problem, the proposed methodology applies a Boolean Satisfiability approach in order to allocate a transistor network with any structure in a minimum width 1D layout. The placement problem is described into a set of variables and clauses according to pre-established placement constraints. These clauses are computed by a SAT solver that defines whether exists a set of variable assignment that satisfies all the clauses.

In order to analyze the results, we compare our placement solution with the layouts generated by the current ASTRAN placement technique. The following aspects are considered in this comparison: area, total wirelength (metal 1 and polysilicon) and contacts. Our solution presented an equal or smaller area in 88.68% of the cases, with 3.02% optimization. An overhead of 0.82% was obtained in the total wirelength. Finally, the number of contacts reduced 12.05%.

In future works, we intend to investigate other parameters of the cell (such as power consumption and delay) in order to have an accurate evaluation of our proposition.

Acknowledgments

Research partially supported by Brazilian funding agencies CAPES, CNPq and Fapergs.

References

[1] Reis, A. "Design Automation of Transistor Networks, a new Challenge," Circuits and Systems (ISCAS), 2011 IEEE International Symposium on,(Jul. 2011)., pp.2485-2488

[2] T. Uehara and W. M. vanCleemput, "Optimal Layout of CMOS Functional Arrays," in IEEE Transactions on Computers, Vol. C-30, No. 5, pp-305-312, May 1979.

[3] J-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishbum and A. E. Dunlop, "Optimization-based transistor sizing," IEEE Journal of Solid-state Circuits, Vol. 23, No. 2, pp.400-409, April 1988.

[4] A. Gupta, S-C. The and J. P. Hayes, "XPRESS: A Cell Layout Generator with Integrated Transistor Folding," In European Design and Test Conference (Mar. 1996), 393-400.

[5] B. S. Carlson, "Transistor chaining and transistor reordering in the design of CMOS complex gates", Ph.D. Dissertation, Syracuse University, August 1991.

[6] T. Iizuka, M. Ikeda and K. Asada, "High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability," ASP-DAC '04: Proceedings of the 2004 conference on Asia South Pacific design automation, 2004, pp. 149-154.

[7] T. Iizuka, M. Ikeda and K. Asada, "Exact Minimum-Width Transistor Placement for Dual and Non-dual CMOS cells," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2005, pp. 3485-3491.

[8] D. D. Hill, "Sc2: A hybrid automatic layout system," Proc.IEEE Int. Conf. on Computer-Aided Design, pp. 172-174, Nov. 1985.

[9] C. Y. Hwang, Y-L. Lin and Y-C. Hsu, "LiB: A CMOS cell compiler,“ IEEE Trans. on Computer-Aided Design, Vol.10, pp. 994-1005, Aug 1991.

[10] A. Ziesemer, C. Lazzari, R. Reis, "Transistor level automatic layout generator for non-complementary CMOS cells," in Very Large Scale Integration, 2007. VLSI - SoC 2007. IFIP International Conference on. USA: IEEE, oct.2007, pp. 116-121.

[11] M. Soos, K. Nohl, and C. Castelluccia, "Extending SAT Solvers to Cryptographic Problems," In proc. of SAT, LNCS 5584, pp. 244-257, Springer, 2009.

[12] Federal Univ. Rio Grande do Sul, Logics Lab. (Oct. 2012). Catalog of 53 Handmade Optimum Switch Networks.[Online]. Available: http://www.inf.ufrgs.br/logics/docman/53_NSP_Catalog.pdf

[13] Sutherland, I.; Sproull, B.; Harris. D. Logical Effort: Designing Fast CMOS Circuits. Morgan Kaufmann Publishers, Inc., 1999.

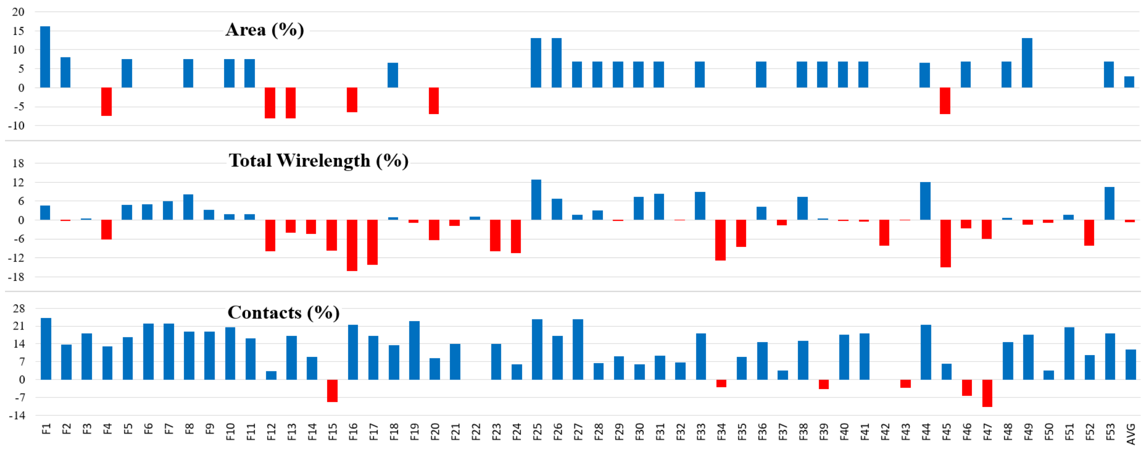

Fig. 3. Geometrical comparatives regarding complex gates cells generated under the proposed and the original placement methodologies implemented into ASTRAN CAD tool.