Transistor Placement for Automatic Cell Synthesis Through Boolean Satisfiability

Maicon Cardoso*, Andrei Bubolz*, Jordi Cortadella[†], Leomar Rosa Jr.*, Felipe Marques*
*Graduate Program in Computing, Federal University of Pelotas, Pelotas, Brazil
[†]Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain
{mscardoso, aaobubolz, leomarjr, felipem}@inf.ufpel.edu.br*, jordi.cortadella@upc.edu[†]

***Abstract*—This paper presents a new transistor placement method applied to the ASTRAN EDA tool, an open-source solution for the automatic design of complex digital gates. Although it currently reaches an optimized solution through a Threshold Accepting approach, ASTRAN does not guarantee a minimum-width placement. In this paper, a method based on Boolean satisfiability is proposed, ensuring an optimal solution for the transistor placement task through modeling the problem into a set of Boolean variables and clauses aware of four design rule constraints. Experiments comparing the proposed method and the current ASTRAN placement technique have shown reductions in the layout area. Furthermore, our method achieved a significant improvement regarding runtime, an essential feature for designing digital circuits and systems on-demand.**

***Keywords*—transistor placement, transistor chaining, Boolean satisfiability, SAT, ASTRAN, EDA tool**

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 significant improvements in different aspects of the digital design [1-4]. In this scenario, the transistor placement procedure is an essential step of the on-demand gate design flow since it enables to achieve optimized layouts concerning several attributes such as area, power, and delay.

The first placement method [5] proposed the width minimization of dual networks through a graph-based approach which aims to find a solution with the minimum number of diffusion breaks. It also presented the widely-used single-row-height layout style, where the transistors are placed exclusively in one direction along two P/N diffusion rows.

Following this, since logic gates should be designed to fulfill some requirements, transistor sizing and folding must be efficiently applied for the synthesis tools. In this scenario, [6] proposed a method to generate layouts without any structure constraints, producing cells with a different number of P and N transistors and non-uniform transistor widths (by applying transistor folding), for instance. Furthermore, still addressing the width minimization of dual circuits, [7] proposed a solution for this problem considering the case of non-series-parallel (NSP) planar circuits through a heuristic-based approach.

Although most of these methodologies produces optimized results, they do not guarantee the minimum width placement of the transistors, i.e., the optimal solution in terms of area. Recently, [8-9] proposed a Boolean satisfiability (SAT) approach introducing an optimal placement method for both dual [8] and non-dual transistor networks [9]. The technique consists in to describe the placement problem into a set of Boolean expressions (clauses) and determines whether there is some variables assignment that satisfies all these clauses.

Along with that, several solutions were proposed integrating placement methods with a complete cell design flow [10-12]. These electronic design automation (EDA) tools use different techniques for placement in order to produce optimized layouts for dual and non-dual logic networks. For instance, while Layout Synthesizer [10] heuristically pairs transistors to be placed, LiB [11] identifies strongly connected clusters and form pairs of each cluster, a more scalable approach. ASTRAN [12], an open-source EDA tool for complex gate design, applies a Threshold Accepting (TA) method [13] to determine the placement solution which maximizes the diffusion sharing and minimizes the interconnection length. This methodology is based on a local search algorithm that starts from a random feasible placement and then explores the solution space looking for a better layout. However, as expected for a TA approach, the placement solution eventually gets stuck in a local minimum, not reaching the optimal transistor ordering in terms of the layout area.

The transistor placement methodology presented in this paper aims to avoid this problem in ASTRAN by using a Boolean satisfiability approach similar to the previously proposed in [8], but focusing on the single-row layout style without any topological constraints, i.e., including non-dual and non-planar arrangements. This way, our proposition is able to deal with CMOS 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. The ASTRAN EDA Tool

ASTRAN is an open-source EDA tool that was developed to automate the physical synthesis of VLSI circuits [12]. This tool automatically implements digital gates at the layout level in 65nm and 45nm technology nodes. It supports cells with different characteristics, such as a wide range of transistor arrangements, unrestricted circuit structures (including non-dual and non-planar topologies), conditional design rules, transistor folding, among others.

The input file of ASTRAN is a transistor network description in SPICE file format. The output layout files of the tool can be written in CIF and GDSII formats. These files may be used in other tools to realize further physical synthesis steps

The authors thank FAPERGS, CNPq, and CAPES (Finance Code 001) Brazilian research support agencies that financed this investigation.

or to perform the cell characterization. It is important to notice that ASTRAN only generates single-row-height layouts.

Regarding the placement procedure currently working on ASTRAN, a Threshold Accepting (TA) approach is implemented where its cost function f_C takes into account parameters relative to the transistor chaining and the detailed routing (to be performed after the placement of the transistors). This cost function f_C is presented in (1),

$$f_C = 100(4W_{gm} + 4W_{md} + 3W_c + 2W_g + W_{wl}) + W_{ld} \qquad (1)$$

where W_{gm} is the weight of the gate mismatch, W_{md} is related to the routing density, W_c refers to the cell width (in terms of the number of elements placed in a matrix data structure), W_g is the weight associated with the number of diffusion gaps, W_{wl} represents the total length of the connections, and, finally, W_{ld} is the local routing density. Along with that, the perturbation function incrementally modifies the initial placement (obtained randomly) by moving a set of contiguous P/N transistors through its diffusion rows. These moves can be done by shifting the positioning of the transistors or flipping its terminals. This way, the cost function measures the quality of the new solutions, accepting or rejecting it accordingly with the current threshold value. After some iterations, it is expected that a good solution (considering its score relative to f_C) has been achieved.

III. Proposed Methodology

This section will describe all the steps of the proposed placement methodology. It is important to mention that our method is similar to the one described in [8]. However, in our approach we focused on obtaining an optimized SAT-based solution (in terms of number of variables and clauses) for the specific layouts generated by ASTRAN, i.e., without topological constraints (including non-dual and non-planar networks) – a necessary constraint when dealing with complex logic gates generated on-demand [4] – and for the specific single-row-height layout style. Our approach also avoids any vertical gate mismatches for the reasons reported in [14].

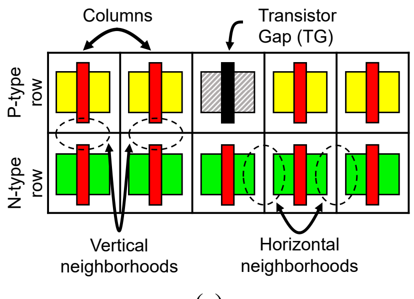
A. Problem Description

The input of the proposed method is a transistor-level description of the complex gate, i.e., a SPICE netlist containing the arrangements of the pull-up (PU) and pull-down (PD) transistors. As proposed originally in [5], in the single-row layout style adopted (Fig. 1.a) the transistors are placed into two rows containing the PU (upper row) and the PD (bottom row) devices. In this scenario, each device is expressed by a set of Boolean 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 input netlist.

For modeling the transistor placement problem, we generate the set of SAT clauses through five main steps. The

TABLE I. Sets of Boolean Variables of the Proposed Modeling

Name	Condition that assigns the variable *true*	Initial 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*	$2N^2$



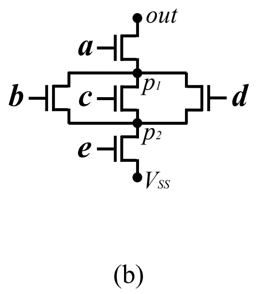


Fig. 1. Instances of a single-row layout (a) and an NMOS arrangement (b).

first one is a simplification procedure (discussed in section III.C), while the remaining implement the placement constraints (section III.D-III.G). Finally, the complete algorithm for placement is presented in Algorithm 1 (section III.H).

B. Transistor Gap (TG)

This component operates like a gap (break in the diffusion row) in the proposed placement modeling. Diverging from the conventional transistor, it might assume dynamic values in its terminals, allowing it 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 transistor gap (TG) may be employed: (1) in case of a different number of P and N-type transistors in the input netlist, where the necessary number of TGs is added to complete the equality; (2) after each iteration resulting in an unsatisfiable placement, in which a TG 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 recreates variables and clauses in order to proceed with the computation. Fig. 1.a shows an instance of a TG used for balancing the number of transistors in each row.

C. Variables Reduction

This step is executed only in the first iteration of the placement algorithm (Algorithm 1), where at least one row is free of gaps. The goal of this procedure is the reduction of the number of variables and clauses in the modeling. This is done through the analysis of each potential position where a transistor can be placed. First, the source and drain identifiers of all the transistors of are stored. Then, for the terminals that appear only once, defined as unique terminals, two possible scenarios are taken: (1) in case of none unique terminal, then there is nothing to be simplified and the algorithm proceeds; (2) in case of one or two unique terminals, then the transistors which contain these unique terminals cannot be placed between other devices, i.e., it should be placed at the boundaries of the cell or beside TGs.

To exemplify the impacts of this optimization in the placement modeling, consider the NMOS transistor network illustrated in Fig. 1.b, where we can see that two of its internal nodes (p_1 and p_2) are shared by all the switches. On the other hand, the external nodes (*out* and V_{SS}) appear only once, i.e., they are unique terminals of this network. In this scenario, through the procedure described above, the number of Boolean variables decreases from 100 to 72, while the number of clauses reduces from 3797 to 2525, an optimization of 28.0% and 33.5% in these parameters, respectively.

D. *Transistor Allocation Constraint*

Each transistor must be allocated exclusively in one cell location. The clauses that implement this constraint for the PU row are created through the formula (2) presented below,

$$\forall i \in \{1,\dots,P\}, \forall c \in \{1,\dots,C\}, \quad T_P(i,c) \Rightarrow \bigwedge_{\substack{\forall c' \in \{1,\dots,C\},\\ c' \neq c}} \neg T_P(i,c') \tag{2}$$

where P is the number of devices (including TGs) in the PU network, C is the number of columns in the current iteration, and

$$T_P(i,c) \equiv T_P(i,c,f) \lor T_P(i,c,\overline{f}) \tag{3}$$

such that f and $\overline{f}$ denotes the two possible ordering – normal or flipped, respectively – of the source and drain terminals of T_P accordingly with the input netlist. A similar formula also holds for the PD transistor network.

E. *Empty Column Constraint*

Each position of the cell should contain at least one device (transistor or TG). The clauses that implement this constraint for the PU transistor arrangement are presented in (4),

$$\forall c \in \{1,\dots,C\}, \quad \bigvee_{\forall i \in \{1,\dots,P\}} T_P(i,c) \tag{4}$$

where a similar formula is also applied to the PD arrangement.

F. *Diffusion Sharing Constraint*

All transistors must have its lateral neighbors with equivalent source and drain values, except for those placed in the boundaries (which only need one of these). Moreover, it is important to notice that the gaps, represented by TGs, may be connected to any other device since it can assume dynamic values for its drain and source terminals.

To implement the constraint that avoids neighbors sharing divergent terminals, (5) is applied for each pair of transistors $T_P(i, c)$ and $T_P(j, c+1)$ – not including TGs in this set.

$$\begin{array}{l}\forall i \in \{1,\dots,P\},\\ \forall j \in \{1,\dots,N\},\\ \forall c \in \{1,\dots,C\},\end{array} \begin{cases} \neg\left(T_P(i,c,f) \land T_P(j,c+1,f)\right) \text{ if } \mathrm{s}(i) \neq \mathrm{d}(j), \\ \neg\left(T_P(i,c,f) \land T_P(j,c+1,\overline{f})\right) \text{ if } \mathrm{s}(i) \neq \mathrm{d}(j), \\ \neg\left(T_P(i,c,\overline{f}) \land T_P(j,c+1,f)\right) \text{ if } \mathrm{s}(i) \neq \mathrm{d}(j), \\ \neg\left(T_P(i,c,\overline{f}) \land T_P(j,c+1,\overline{f})\right) \text{ if } \mathrm{s}(i) \neq \mathrm{d}(j) \end{cases} \tag{5}$$

In formula (5), s(i) is the node connected to the source terminal of the transistor $T_P(i, c)$ in its orientation f (normal or flipped) accordingly with the input netlist and d(j) is the node connected to the drain terminal of $T_P(j, c+1)$ also respectively to its orientation f in the netlist. A similar formula is applied to restrict the diffusion sharing of the PD transistor arrangement.

G. *Gate Alignment Constraint*

All P and N-type transistors placed in the same column (in its respective rows) must share their gate signals. In other words, if there is a transistor $T_P(i, c)$ placed in the top row in column c, then a transistor $T_N(j, c)$ controlled by the same gate input must also be placed in c in the bottom row. As before, the gaps represented by TGs can be associated with any gate value. The clauses that implement the constraint responsible for avoiding gate mismatches are modeled through (6), in which TGs are not included in the devices sets – as previously in (5).

$$\begin{array}{l}\forall i \in \{1,\dots,P\},\\ \forall j \in \{1,\dots,N\},\end{array} \quad \bigwedge_{\forall c \in \{1,\dots,C\}} \neg\left(T_P(i,c) \land T_N(j,c)\right) \text{ if } \mathrm{g}(i) \neq \mathrm{g}(j) \tag{6}$$

In (6), g(i) and g(j) are the gate signals responsible for controlling the transistors $T_P(i, c)$ and $T_N(j, c)$, respectively.

H. *Placement Algorithm*

The optimal SAT-based placement method proposed is presented in Algorithm 1 and it was implemented in ASTRAN.

Algorithm 1 Pseudocode of the SAT-based Placement Method

```
1:  placeTransistors ( netlist N )
2:    minCols ← getColumnsLowerBound ( N )
3:    trList ← createTransistorsList ( minCols, N ) ∪ createVariableTransistors ( N )
4:    isSAT ← false
5:    while not isSAT do
6:      V ← createBooleanVariables ( trList )
7:      C ← createClauses ( V )
8:      if isSatisfiable ( V, C ) then
9:        validAssignment ← SAT ( V, C )
10:       isSAT ← true
11:     else then
12:       trList ← addVariableTransistors ( trList )
13:     end if
14:   end while
15:   return decodeSolution ( trList, validAssignment )
16: end
```

The algorithm receives as its input the netlist N in SPICE format (line 1) and computes the minimal number of columns (lower bound) as described in [15] (line 2). After, the *trList* is created through the variable reduction step described previously (section III.C) with the addition of the transistor gaps generated to make the number of devices of each plan even (line 3). Following this, the Boolean variables V and the clauses C are created (lines 6 and 7, respectively), where C is the conjunction of the formulas (2) and (4-6) in conjunctive normal form (CNF). In case of satisfiability, the satisfiable assignment of the Boolean variables is computed (line 9). On the other hand, if the formula is not satisfiable, then one transistor gap is added on each logic plan (line 12) – this process is equivalent to increasing the number of columns by one on the search space, an approach that ensures the minimum area. Finally, after finding a SAT solution, a decoding function is executed in order to transform the variables assignment that evaluates the clauses to true for the placement format to be computed by ASTRAN (line 15).

IV. Results

This section presents a comparison of the layouts produced by the proposed method and the ASTRAN placement approach.

The first experiment consists of measuring the number of columns in the pseudo-layouts (as the 5-column one illustrated in Fig. 1.a) produced through each approach. In this scenario, the benchmarks used in this experiment are the following: the 53 NSP handmade networks (53NSP) [16] and a subset of the 4-input P-class (P4) [17]. While the former is composed by 53 handcrafted non-series-parallel gates containing from 10 to 14 transistors, the latter is a subset of 150 cells generated through the Kernel Finder algorithm [18]. To choose the networks to be part of the assessed subset, we divided the catalog into 10 classes accordingly to the number of transistors in each complex gate arrangement (ranging fromddd 6 to 24 transistors). For each one of these classes, we randomly selected 15 cells to be implemented by both approaches. This way, we guaranteed a homogeneous distribution regarding the size of the netlists.

The results of this first assessment are presented in Table II, which consists in a frequency table of the differences in the number of columns of the layouts where the reference is the ASTRAN original solution (based on TA). Notice that, for the 53NSP catalog, 73.6% of the cells presented the same number of columns, while 26.4% of the pseudo-layouts reported optimizations in this aspect. Regarding the P4 benchmark, 77.3% of the solutions presented the same number of columns, while 22.7% reported reductions on this attribute. It is important to notice that the SAT solution does not present any overhead relative to the TA approach since it guarantees an optimal layout in terms of the number of columns.

The second experiment was conducted under the 53NSP catalog by generating the layouts through the complete flow provided by the ASTRAN tool. In this scenario, we employ the original routing and compaction procedures of ASTRAN for both versions of the layouts generated. Moreover, the STMicroelectronics 65nm node was adopted for the synthesis with all the ASTRAN's options in default. As SAT solver, we used the CryptoMiniSat [19] also on its default configuration.

The following parameters were extracted from the layouts: area, routing wirelength (considering polysilicon and metal 1), and number of contacts. Besides that, the execution times of the placement procedures were also collected. Fig. 2 shows the comparison, where we take as reference the original TA approach. For the layout area, an optimization of 2.0% was obtained; regarding wirelength, an overhead of 0.1% was observed; related to the number of contacts, the SAT-based solution presented 5.9% less contacts; finally, the proposed approach took 0.49s on average to compute the placement, while the original ASTRAN procedure spent 11.42s on average for the same scenarios, thus corresponding to a 23.3x speedup. Finally, the standard deviations of the optimizations in area, wirelength, number of contacts, and runtime are 3.8%, 8.3%, 6.1%, and 4.5%, respectively. Through the analysis of Fig. 2 along with the corresponding standard deviation of each parameter, we can notice a small dispersion of the data, i.e, the results are uniform for most of the cases.

TABLE II. Frequency Table of the Difference on the Number of Columns of the Layout Considering the TA Approach as Reference

Difference (# columns)	Benchmarks	
	***53 NSP handmade networks* [16]**	***Subset of the 4-input P-class* [17]**
0	39 (73.6%)	116 (77.3%)
-1	13 (24.5%)	25 (16.7%)
-2	1 (1.9%)	7 (4.7%)
-3	0 (0.0%)	2 (1.3%)

Finally, we can notice that, even with the minimal placement provided by the SAT approach (as shown in Table II), the layout area of two cells (F16 and F20) presented an overhead compared to the original ASTRAN version. This is due to the routing step, which uses extra columns to perform the internal connections of the gate.

V. Conclusions

This paper presented a new Boolean satisfiability-based placement method applied to the ASTRAN EDA tool, an open-source solution to design circuits and systems on-demand that currently implements a Threshold Accepting algorithm as its transistor placement procedure. The main difference between our technique and the original one is that we guarantee an optimal cell regarding the number of layout columns.

As reported in the results section of this paper, the proposed placement solution was capable to deliver optimized cells concerning the number of columns used for placement, total area, and number of contacts, while presenting a small overhead in wirelength. Besides that, in terms of runtime, our algorithm was considerably faster compared to the currently implemented in ASTRAN, an important feature considering the on-demand design – the main propose of the tool.

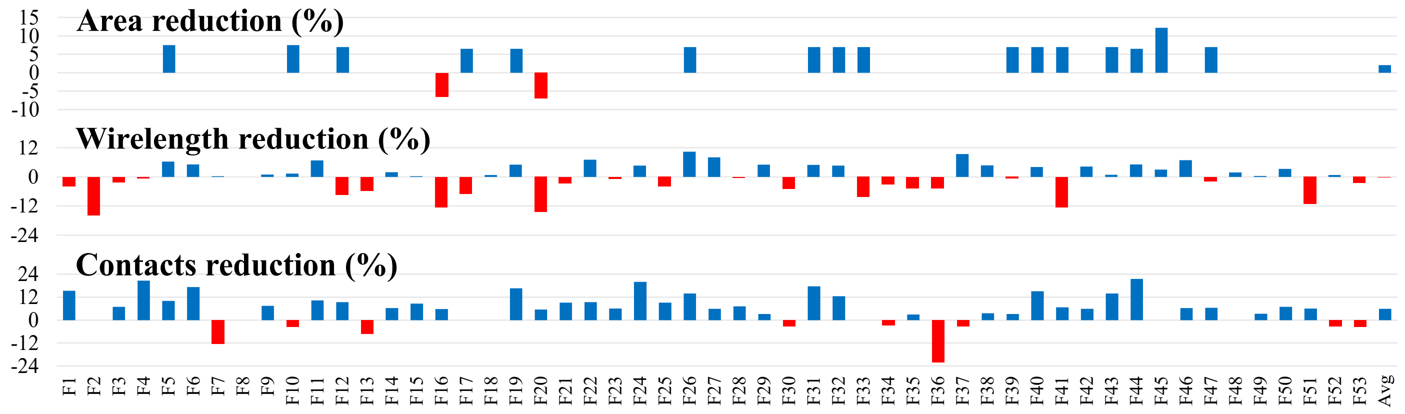


Fig. 2. Optimizations and overheads of the layouts produced for the catalog of 53 NSP handmade networks [16] obtained through the proposed Boolean satisfiability-based placement approach. The reference is the original ASTRAN placement method based on Threshold Accepting.

REFERENCES

[1] R. Reis, "Design automation of transistor networks, a new challenge," in *IEEE International Symposium of Circuits and Systems (ISCAS)*, Rio de Janeiro, Brazil, 2011, pp. 2485–2488.

[2] M. Cardoso, G. Smaniotto, R. Zanandrea, R. Souza, L. Rosa, and F. Marques, "Physical design of supergate cells aiming geometrical optimizations," in *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Abu Dhabi, United Arab Emirates, 2016, pp. 1–4.

[3] C. Conceição and R. Reis, "Transistor count reduction by gate merging," *IEEE Transanctions on Circuits and Systems I: Regular Papers*, vol. 66, no. 6, pp. 2175–2187, 2019.

[4] M. Cardoso, G. Smaniotto, A. Bubolz, M. Moreira, L. Rosa, and F. Marques, "Libra: an automatic design methodology for CMOS complex gates," *EEE Transanctions on Circuits and Systems II: Express Briefs*, vol. 65, no. 10, pp. 1345–1349, 2018.

[5] T. Uehara and W. VanCleemput, "Optimal layout of CMOS functional arrays," *IEEE Transactions on Computers*, vol. C–30, no. 5, pp. 305–312, 1981.

[6] A. Gupta, S. The, and J. Hayes, "XPRESS: A cell layout generator with integrated transistor folding," in *European Design and Test Conference (ED&TC)*, Paris, France, 1996, pp. 393–400.

[7] B. Carlson, "Transistor chaining and transistor reordering in the design of CMOS complex gates," Ph.D dissertation, Syracuse University, 1992.

[8] T. Iizuka, M. Ikeda, and K. Asada, "High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Yohohama, Japan, 2004, vol. E87-A, no. 12, pp. 3293–3300.

[9] T. Iizuka, M. Ikeda, and K. Asada, "Exact minimum-width multi-row transistor placement," in *International Symposium on Circuits and Systems (ISCAS)*, Island of Kos, Greece, 2006, pp. 5431–5434.

[10] D. Hill, "Sc2: a hybrid automatic layout system," in *IEEE International Conference on Computer-Aided Design (ICCAD)*, 1985, pp. 172–174.

[11] Y. Hsich, C. Hwang, Y. Lin, and Y. Hsu, "LiB: a CMOS cell compiler," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 8, pp. 994–1005, 1991.

[12] A. Ziesemer and R. Reis, "Physical design automation of transistor networks," *Microelectronic Engineering*, vol. 148, pp. 122–128, 2015.

[13] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of Computational Physics*, vol. 90, no. 1, pp. 161–175, 1990.

[14] M. Cardoso, G. Smaniotto, J. Machado, M. Moreira, L. Rosa, and F. Marques, "Transistor placement strategies for non-series-parallel cells," in *Midwest Symposium on Circuits and Systems (MWSCAS)*, Boston, USA, 2017, pp. 523–526.

[15] J. Cortadella, "Area-Optimal Transistor Folding for 1-D Gridded Cell Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 11, pp. 1708–1721, 2013.

[16] Logics Lab, *Catalog of 53 handmade optimum switch networks*, Federal University of Rio Grande do Sul, 2012. Acessed on: Oct. 2019. [Online]. Available: http://www.inf.ufrgs.br/logics/docman/53_NSP_Catalog.pdf.

[17] V. Correia and A. Reis, "Classifying n-input Boolean functions," in *Workshop IBERCHIP*, 2001, pp. 58–66.

[18] V. Possani, V. Callegaro, A. Reis, R. Ribas, F. Marques, and L. Rosa, "Graph-based transistor network generation method for supergate design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 2, pp. 692–705, 2015.

[19] M. Soos, K. Nohl, and C. Castelluccia, "Extending SAT solvers to cryptographic problems," in *Theory and Applications of Satisfiability Testing (SAT)*, Swansea, UK, 2009, pp. 244–257.