SATSoT: A Methodology to Map Controllable-Polarity Devices on a Regular Fabric Using SAT

Catherine Gasnier, Pierre-Emmanuel Gaillardon, Giovanni De Micheli
Integrated Systems Laboratory (LSI), EPFL, Switzerland
Email: catherine.gasnier@epfl.ch

***Abstract*—Devices with controllable-polarity, such as *Double-Gate Vertically-Stacked Nanowire FETs*, have shown promising interests in recent years to implement XOR-based logic functions in an unprecedented compact way. Such a compactness is obtained at the cost of a denser interconnect, that can be mitigated by designing an efficient hyper-regular layout structure, called *Sea-of-Tiles*. In this paper, we propose a methodology, based on Boolean satisfiability, to map netlists of transistors on such a structure. The methodology endeavors to minimize the wiring complexity, by maximizing the sharing of the different terminals. We showed that its implementation, *SATSoT*, is able to automatically generate compact mappings with wiring complexities similar to manual layouts.**

I. Introduction

Multiple Gate Field-Effect Transistors (MuGFET) offer promising opportunities to further scale down the transistor sizes, by reducing leakage current and enhancing the drive current [1]. FinFETs are for example scalable down to 20nm [2]. Among those MuGFETs, peculiar devices present a controllable polarity, i.e., are able to exhibit either *n*-type or *p*-type characteristics, depending on the polarity applied on a second independent gate. This is the case of *Vertically-Stacked Silicon Nanowire Field Effect Transistors* (SiNWFETs) [3], Carbon Nanotube Field Effect Transistors (CNFETs) [4], and Graphene Field Effect Transistors [5].

Thanks to their two independent gate structures, those devices have a higher expressiveness than conventional CMOS devices, i.e., it is possible to implement more complex Boolean functions with fewer resources. In particular, the XOR function can be implemented in a very compact way [6]. However, the additional gate on every device is expected to significantly increase the wiring complexity. To mitigate this effect, a specific layout methodology has been introduced in [8]. This methodology relies on the use of a regular organization, called *Sea-of-Tiles* (SoT), that can be efficiently realized by SiNWFETs. Regular fabrics, such as SoT, enable a manufacturability improvement of ICs at small technology nodes [7]. In [8], the mapping of logic gates on SoT was proposed based on a traditional standard-cell design approach. However, the discrete nature of this problem makes it suitable for being solved by Boolean satisfiability techniques.

The Boolean satisfiability problem, also abbreviated as SAT, is the first known NP-complete problem [9]. Many efficient algorithms for SAT have been designed and implemented in a plethora of SAT solvers (GRASP [10], Minisat [12], ppfolio [13], among many others). Relying on this algorithmic efficiency, methodologies involving SAT have been proposed for many difficult problems in CAD, such as channel-routing and placement [14], or routing of FPGAs [15].

In this paper, we propose to automatize the mapping of short netlists of transistors onto SoT structure using Boolean satisfiability. We target netlists of a few tens of transistors, which is the size of typical logic gates. Automating the mapping of logic gates onto regular fabrics enables to quickly build large standard cell libraries, while keeping the wiring complexity under control. Such an opportunity is of paramount importance to better exploit the expressiveness of the transistors at the gate level. Mapping results show that the presented methodology allows us to automatically exploit the sharing of all the different terminals of controllable-polarity transistors, leveraging a good implementation compactness of logic cells. Indeed, the methodology achieves wiring complexities similar to mappings obtained manually.

The paper is organized as follows. Section II introduces the required background on controllable-polarity devices and Boolean satisfiability in more details. Section III depicts the way we use satisfiability to solve the aforementioned problems. Finally, Section IV presents the performance of our implementation and Section V draws some conclusions.

II. Background

In this section, we describe, in more details, controllable-polarity devices and their associated layout methodology. Then, we introduce the necessary background on Boolean satisfiability.

A. Compact Logic Gates with Controllable-Polarity Devices

A controllable-polarity device is, in general an ambipolar field-effect transistor with double-gate structure, whose polarity (*n*- or *p*-type) is controlled by the bias voltage applied on the second gate terminal, called *polarity gate*. As illustrated in Fig. 1a, if the polarity gate has a high (respectively low) bias voltage, the transistor behaves as *n*-type (respectively *p*-type).

Such a conduction property inherently implements the XNOR logic function between the two gate logic levels. For that reason, it is possible to design full-swing XOR gates using only four such transistors, as shown in Fig. 1b [6]. While *binate* logic functions (such as XOR/XNOR) take full-advantage of the controllable-polarity transistor properties, the design of usual CMOS gates (namely *negative-unate* Boolean functions such as NAND/NOR) is achieved by biasing the polarity gates of transistors to either V_{DD} or Gnd to obtain traditional unipolar transistors. As an example, the circuit of a NAND gate is depicted in Fig. 1c.

Among the different technology competitors, *Double-Gate Silicon Nanowire Field-Effect Transistors* (DG-SiNWFETs) are very promising to realize controllable-polarity devices [3]. Their structure and abstracted top view are shown in Fig. 2, as introduced in [8].

Devices with controllable-polarity introduce an additional terminal to route on every single transistor, leading to an increased wiring complexity in the first metal layers. To mitigate this impact and keep the wiring under control, a symbolic-layout diagram, called *dumbbell-stick diagrams*, and an associated layout methodology were

978-1-4799-0874-5/13/$31.00

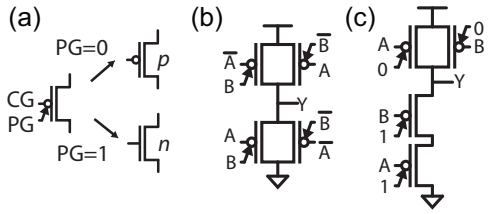


Fig. 1. (a) Symbol of a controllable-polarity transistor. (b) Complementary XOR logic gate exploiting the higher expressiveness of controllable-polarity transistors. (c) Complementary NAND logic gate with polarity gates biased to either V_{DD} or Gnd.

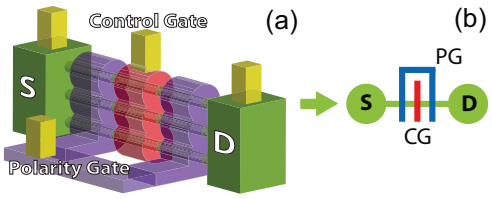


Fig. 2. (a) 3D view of a DG-SiNWFET. (b) Abstracted top view.

introduced in [8]. Instead of focusing on the pull-up and pull-down parts of the circuit, the new methodology explores new ways of arranging the devices, considering both the control gate and the polarity gate signals. Therefore, the physical design of logic functions exploiting the polarity gates was improved and the wiring kept under control.

In addition, they also presented the concept of *Sea-of-Tiles* (SoT), as an architectural support of the wiring simplification methodology. This rejuvenated *Structured ASIC* methodology is based on elementary building blocks called *Tiles*, that are composed of few transistors sharing their terminals. Note that the concept of SoT can be applied to any controllable-polarity devices. Therefore, the the methodology discussed in the following is not dependent to the considered device. More details about Tiles and SoT will be given in III-A.

B. Satisfiability Solving

The Boolean satisfiability problem, also abbreviated as SAT, is used to efficiently solve a wide variety of problems, especially in *Electronic Design Automation* (EDA): channel-routing and placement [14], routing of FPGAs [15], standard cell routing [16], etc. It benefits from a large panel of solvers [10]–[13] that keep on being improved and compete every year in the international SAT competition [17].

Given a Boolean function f in *Conjunctive Normal Form* (CNF) solving the Boolean satisfiability problem consists of determining whether there exists or not an assignment of its variables so that the function evaluates to *true*. By extension, the problem of finding such an assignment is also referred to as *satisfiability*. A Boolean function in CNF can be interpreted as a set of disjunctive clauses that all need to be satisfied. We call a SAT instance a set of such clauses that define a problem. For example, let us consider the function in CNF $f = (X \vee Y) \wedge (\overline{X} \vee Z \vee \overline{Y})$, where $\wedge$ denote the logical AND function, $\vee$ the OR function and $\overline{X}$ the complement of X. Then this function is satisfied by the assignment $X = 1$, $Y = 0$ and $Z = 1$, since $X = 1$ makes the first clause satisfied while $Z = 1$ makes the second satisfied. For more information about Boolean satisfiability, we refer the reader to [18].

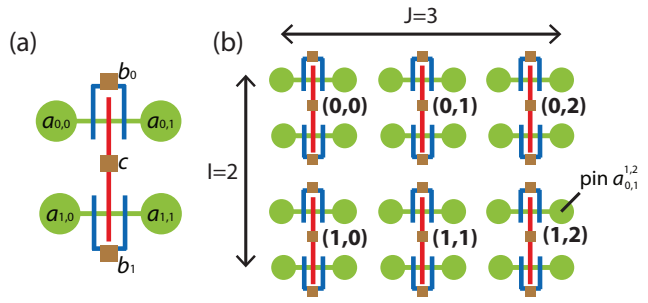


Fig. 3. (a) The basic tile and its 7 pins. (b) An example of a grid of tiles with introduced notations.

C. Motivation

The methodology we propose here automates the mapping of small netlists of transistors onto a set of tiles, enabling to quickly build large gate libraries for many different tile shapes. In particular, it enables to: (i) map any XOR-embedding gate, allowing us to be unrestricted in the elementary Boolean functions used in synthesis; and (ii) impose a given shape on a gate, i.e., restrict its mapping to a certain set of tiles. This way, many different layouts can be constructed for each gate, enabling more flexibility in the placement of the gates.

III. Mapping Transistors on a SoT using SAT

In this section, we describe in details our methodology for mapping a netlist of transistors onto a Sea-of-Tiles using SAT. This methodology targets short netlists, of the typical size of a logic gate.

A. Preliminary Notations

To simplify the instance formulations, we use a very simple tile as depicted in Fig. 3a. The tile, known as Tile G1 [8], consists of two transistors sharing their control gates. It has seven pins in total: one for the shared control gate, two for the polarity gates and four for the drains and sources. Those pins are labelled as in the figure: for each pin, the subscript numbers indicate its position on the tile, while notations a, b and c are used to indicate the type of pin, respectively drain or source pin, polarity gate pin and control gate pin.

For mapping a netlist, we consider a grid of $I \times J$ of such tiles, with I the number of rows and J the number of columns. In the following, we will refer to this grid as the Sea-of-Tiles itself, even if we should rather think of it as a part of the whole Sea-of-Tiles. Such a grid is shown in Fig. 3b, where I is equal to 2 and J equal to 3. In the context of netlist mapping, I and J are determined by the number of transistors to map. These parameters are first set to support the minimal requirements in terms of transistors and a first minimal SAT instance is run. If this instance is unsatisfiable, then the size of the grid is increased until a satisfiable instance is obtained.

An elementary tile is uniquely identified by the indexes (i, j), with $0 \leq i < I$ and $0 \leq j < J$. We note $p^{i,j}$ the pin p from tile (i, j) (for example $a_{0,1}^{i,j}$). Let P be the set of all pins of the SoT and N the set of nets from a given netlist to map.

B. SAT Instance Overview

For the sake of clarity, the various constraints introduced with out SAT instance will be explained by using the example of the mapping

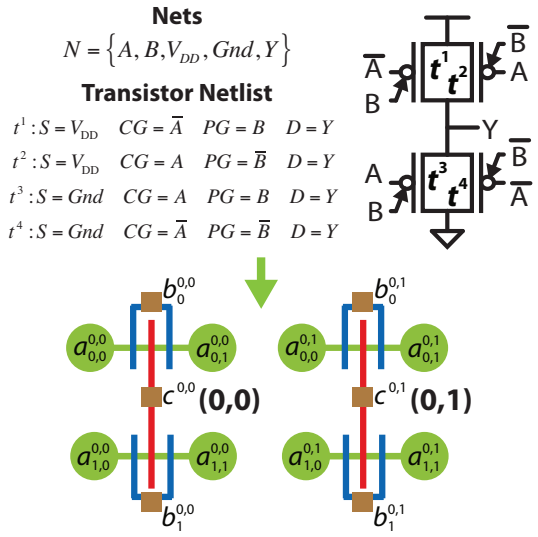


Fig. 4. Problem formulation of a 4-transistor XOR gate mapping.

problem of a 4-transistor 2-input XOR gate onto a 2×2 grid, i.e., the assignment of nets $N = \{A, B, V_{dd}, Gnd, Y\}$ onto the different pins of the SoT. The problem formulation is depicted in Fig. 4.

To solve our mapping problem, we define Boolean variables able to describe a mapping, in the sense that an assignment of those variables, i.e., a solution to our SAT instance, can be directly translated into a valid mapping. A simple way to proceed is to introduce *assignment variables*: if ν is a net of the netlist and p a pin of the SoT, then the assignment variable of net ν to pin p, denoted p_ν, is true if and only if net ν is assigned to pin p. Fig. 5 shows an illustration of an assignment variable on the bottom-left transistor, where $a^{0,0}_{1,0_B} = 1$ means that net B is assigned to the pin $a^{0,0}_{1,0}$.

However, an assignment of those variables does not necessarily correspond to a valid mapping. For example, if both $a^{0,0}_{1,0_A}$ and $a^{0,0}_{1,0_B}$ are assigned to *true*, then both nets A and B have been assigned to the same pin $a^{0,0}_{1,0}$, which is not physical. To avoid that, a first set of constraints, called *validity constraints*, are introduced in III-C.

For a mapping to actually correspond to a given netlist, we need to consider the position of the transistors in addition to the position of the nets. Therefore, we define, in III-D, a set of *transistor assignment variables* in a similar way than nets. Again, we need constraints to ensure that each transistor of the netlist is placed at exactly one position on the SoT, and that no transistors are overlapped. Therefore, we define a set of *placement constraints*. These constraints are then derived in terms of assignment variables, as we expect that a valid assignment of the transistors turns into a valid assignment of the nets.

The proposed set of constraints is enough to guarantee a valid mapping of the netlist. However, some efforts are required to minimize the wiring complexity. To give to the SAT solver a sense of what a good mapping is, we introduce a set of *optimization constraints*. Those are detailed in III-E.

C. Validity Constraints

As introduced above, pin assignment variables can describe valid mappings but they are not necessarily guaranteeing that an assignment of those variables corresponds to a valid mapping. Therefore, we

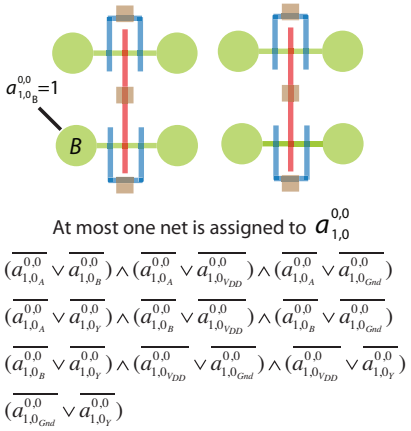


Fig. 5. Illustration of assignment variables and validity constraints for pin $a^{0,0}_{1,0}$.

define a set of *validity constraints*: if net ν is assigned to pin p, then for each net μ different from ν, μ is not assigned to p. This results in Boolean notation in:

$$\forall p \in P, \bigwedge_{\nu \in N} \bigwedge_{\mu \neq \nu \in N} (p_\nu \Rightarrow \overline{p_\mu}) \tag{1}$$

where the symbol $\Rightarrow$ denote the Boolean implication. These *validity constraints* force any assignment of Boolean assignment variables to correspond to valid mappings. From (1), we can easily obtain a CNF using the identity $P \Rightarrow Q = \overline{P} \vee Q$:

$$\forall p \in P, \bigwedge_{\nu \in N} \bigwedge_{\mu \neq \nu \in N} (\overline{p_\nu} \vee \overline{p_\mu}) \tag{2}$$

Fig. 5 also illustrates the validity contraints associated with pin $a^{0,0}_{1,0}$, by listing all the possibles couples on assignments required to ensure a unique assignment.

D. Transistors Placement Constraints

In addition to the considerations on nets, the validity of a mapping implies that each transistor of the netlist is assigned to a unique transistor of the SoT. Hence, we define a set of *transistor assignment variables* in a similar way than pin assignment variables: each transistor assignment variable determines whether or not a given transistor t of the netlist is assigned to a given transistor of the SoT.

Given integers i and j such that $0 \leq i < I$ and $0 \leq j < J$, let $(i, j, 0)$ (respectively $(i, j, 1)$) denote the position on the SoT of the upper (respectively lower) transistor of tile (i, j). Let then G be the domain of (i, j, k), i.e., $\{0, ..., I-1\} \times \{0, ..., J-1\} \times \{0, 1\}$. Then, given a transistor t of the netlist and (i, j, k) in G, we denote by $t_{i,j,k}$ the transistor assignment variable of transistor t to the transistor of the SoT at position (i, j, k), i.e., $t_{i,j,k}$ is true if and only if transistor t is placed at position (i, j, k). An example of such a transistor assignment variable is given in Fig. 6 with the assignment of the transistor instance t^3 to the top-right transistor.

Each transistor t is assigned to at least one of the transistors of the SoT, i.e., for a transistor t of the netlist, at least one of the $t_{i,j,k}$ variables is true. This can be written in Boolean notation with

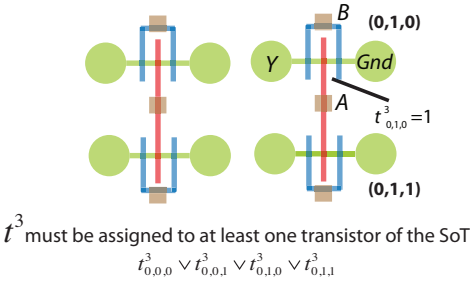


Fig. 6. Illustration of transistor assignment variables and associated placement constraint for t^1.

a disjunction of the $t_{i,j,k}$, as follows:

$$\bigwedge_{t \in T} \bigvee_{(i,j,k) \in G} t_{i,j,k} \tag{3}$$

Figure 6 gives also an example of such a constraint associated with transistor t^3, i.e., t^3 must be placed on one of the four possible transistors of the SoT with indexes$(0,0,0)$, $(0,0,1)$, $(0,1,0)$ or $(0,1,1)$. In addition, no transistor of the SoT should be assigned to more than one transistor instance of the netlist. This is already guaranteed by the constraint, introduced previously, that no pin is assigned to more than one net.

To ensure a valid mapping, a correct assignment of transistors must derive in a pin assignment. Therefore, we introduce new constraints to define those variables in terms of the pin assignment variables. For a transistor t of the netlist, we call A_t and $\widehat{A}_t$ the nets assigned to the drain and source, B_t the one assigned to the polarity gate and C_t to the control gate. For example, in Fig. 7, we would have $A_{t^1} = Y$, $\widehat{A}_{t^1} = V_{\text{DD}}$, $B_{t^1} = B$ and $C_{t^1} = \overline{A}$. Then, a definition of the transistor assignment variables in terms of the assignment variables can be written, in Boolean notation, as follows:

$$\forall t \in T, (i,j,k) \in G \tag{4}$$

$$t_{i,j,k} \Rightarrow {c^{i,j}}_{C_t}$$

$$\wedge t_{i,j,k} \Rightarrow (a^{i,j}_{k,0\,A_t} \vee a^{i,j}_{k,0\,\widehat{A}_t}) \tag{5}$$

$$\wedge (t_{i,j,k} \wedge a^{i,j}_{k,0\,A_t}) \Rightarrow a^{i,j}_{k,1\,\widehat{A}_t} \tag{6}$$

$$\wedge (t_{i,j,k} \wedge a^{i,j}_{k,0\,\widehat{A}_t}) \Rightarrow a^{i,j}_{k,1\,A_t} \tag{7}$$

$$\wedge t_{i,j,k} \Rightarrow {b^{i,j}_k}_{B_t} \tag{8}$$

This expression can be directly converted into CNF, as per (1). An example of such a constraint is presented in Fig. 7. In this example, the formula means that if transistor t^1 is placed at position $(0,0,1)$, then net $\overline{A}$ must be assigned to $c^{0,0}$ (clause (4)) and net B to $b^{0,0}_1$ (clause (8)). The assignment of the drain and source is more complex, because of the interchangeability of the terminals. This liberty is expressed in clauses (5-7). In the example 7, these clauses express the fact that nets Y and V_{DD} can be either associated to pins $a^{0,0}_{1,0}$ or $a^{0,0}_{1,1}$.

E. Optimization Constraints

The constraints introduced so far are sufficient to obtain a correct mapping of the input netlist of transistors. However, our global objective is to minimize the wiring complexity of the mapping, and while different mappings can satisfy the previously defined set of constraints, some solutions might be inefficient regarding the routing

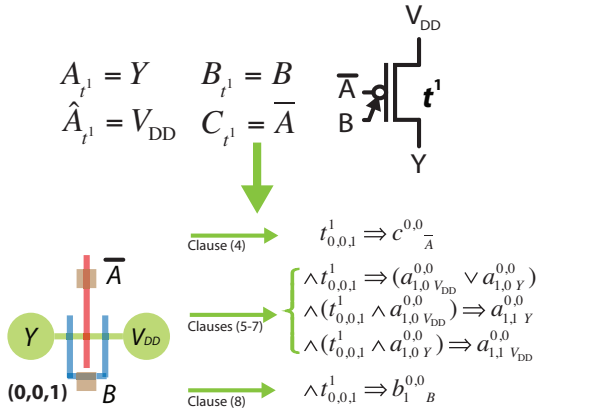


Fig. 7. Expression of transistor assignment variable $t^1_{0,0,1}$ in terms of assignment variables.

complexity, as for example the mapping of Fig. 8a compared to Fig. 8b. While Boolean satisfiability is not fundamentally an optimization problem, we endeavor to optimize the wiring complexity by ensuring that pins that are assigned to the same nets are close to each other.

For that purpose, we formulate a set of constraints to state: if a net ν is assigned to a pin p, then ν must also be assigned to the neighbor pin of p. Applied to pin $b^{i,j}_k$, this constraint can be written in Boolean notation by a simple implication, as follows:

$$\bigwedge_{X \in N} {b^{i,j}_k}_X \Rightarrow {b^{i,j+1}_k}_X \tag{9}$$

for $i \in \{0, ..., I-1\}$, $j \in \{0, ..., J-2\}$ and $k \in \{0,1\}$. Such a constraint is called an *elementary optimization constraint*. By this approach, we do not claim to actually minimize the wiring complexity, since the wiring complexity is not strictly speaking an objective function. Instead, the wiring complexity will be used as a measure of the results quality, as explained in section IV.

There is one elementary optimization constraint per (i,j,k) tuple, i.e., in total, $2 \times I \times (J-1)$ constraints. Hence, enforcing all those optimization constraints may result in unsatisfiable instances. Consequently, a reduced constraint set is determined. The proposed approach consists of imposing first a strong set of constraints and progressively relaxing them until a satisfiable instance is obtained. In practice, the largest number of constraints is often satisfiable, as discussed in section IV, i.e., all optimization constraints can be enforced. Therefore, we start by setting the number of constraints at its maximum value, and decrease it until we find a satisfiable instance. This procedure is first applied to optimization constraints on pins b and then repeated for pins a while having fixed the number of optimization constraints on pins b at its maximum satisfiable value. It has to be noted that this approach only forces pins assigned to the same nets to be next to one another. Enforcing more general rules for efficient routing, such as the avoidance of signal crossing, is out of the scope of this paper.

IV. Mapping Experimental Results

In this section, we present SATSoT, a tool based on the methodology described so far to map netlists of transistors on Sea-of-Tiles using SAT, and show that it achieves satisfying wiring complexity in tractable time.

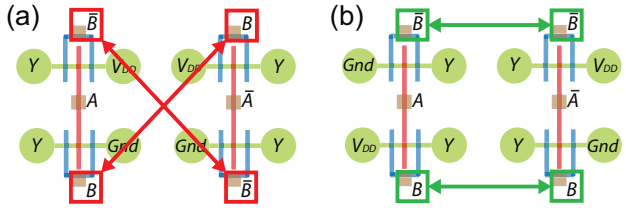


Fig. 8. Two valid mapping of the 4-transistors XOR gate. (a) Mapping with important wiring complexity. (b) Mapping with lower wiring complexity.

A. Methodology and Metrics

SATSoT is implemented using C++ language and works as a wrapper: it generates inputs for a SAT solver and analyses the results. For our experiments, a very simple open-source SAT solver called Minisat [12] was used.

To assess the performance of our methodology, SATSoT is run on two distinct libraries. The first one, *ambilib*, is a library designed specifically for devices with controllable polarity, leveraging their high expressiveness in XOR-embedding gates [6]. In the following, the functions named FXX refer to this library. This library contains 46 gates of up to 3 transistors in series, up to 6 inputs and up to 12 transistors. The second library is a regular CMOS standard-cell library with 84 gates of up to 50 transistors [19]. Mappings were performed on both Tile G1 and Tile G2 architectures. Tile G2 is composed of two Tiles G1, i.e. 4 transistors, sharing their polarity gates and drains and sources [8]. To quantify the generated mappings, we define a measure of the wiring complexity of a mapping as an upper bound of the total length of the wire that will have to be subsequently routed. Considering a mapped net ν and the positions $P_1, ..., P_n$ of its pins, we refer as *wiring complexity of net* ν to:

$$\sum_{i=1}^{n} |P_i - C| \tag{10}$$

where $|\cdot|$ denotes the euclidean distance and C the barycenter of positions $P_1, ..., P_n$. The *wiring complexity of a mapping* is then simply the sum of the wiring complexity of its nets. We report for each mapping its *wiring complexity density*, i.e., its wiring complexity divided by the total area of the mapping. A large wiring complexity density, i.e., important wire lengths per tile, makes a mapping more difficult to wire. Indeed, pins of nets are farther from one another, thereby increasing the probability to have signal crossings inside the gate. In addition, all the wiring complexity metrics are normalized, considering an unitary height of a tile.

B. Experimental Results

Experimental results are shown in Fig. 9. We first illustrates the obtained results with two examples and then we discuss the performances of the methodology considering the two test-case libraries.

1) Methodology Performance Illustration: Fig. 9ab illustrate the mapping of two simple functions on Tiles G1 (F21 [6] and the carry-out logic of a full-adder) obtained by both our methodology and the one introduced in [8]. We compare the equivalent mappings on SoT from [8] with mappings generated by our tool, and show that our automatized methodology is able to achieve similar wiring complexities than the mappings obtained by hand in short runtimes - a few tens of milliseconds. For gate F21, the obtained mappings are exactly the same (Fig. 9a), achieving a wiring complexity of 4.43. The automated mapping was obtained in 28 ms with SATSoT. For the carry-out function, mappings are slightly different with close wiring complexity: 4.17 for the hand-made mapping against 4.50 for SATSoT (Fig. 9b). In this context, the mapping was obtained in 51 ms. In both mappings, polarity gates are nicely grouped together and almost all pairs of neighboring drain-sources are assigned to the same nets. Note that only one such pair could not be mapped on the same net, and is therefore not placed at the same position on both mappings, inducing the slight wiring complexity difference.

2) Global Considerations: Table 9c shows the most significant results from the runs over the two case-study libraries when mapped on Tiles G2. In this table, we report the number of transistors of the gate, the number of tiles used by both our automatized mapping methodology and, when available, by hand [8], the wiring complexity density, the runtime and the number of run instances. The results are sorted by numbers of transistors. Most gates can be mapped with minimal areas, i.e., with a number of Tiles G2 equal to the quarter of the number of transistors, therefore guaranteeing a good usage of the available resources. The number of tiles are equivalent between our solution and the previous art, confirming a good efficiency of the automatized method. The wiring complexity increases roughly linearly with the number of transistors: as the number of tiles increases, the wire length per tile increases due to longer wires. On average over the libraries, the wiring complexity density is 7.35, which means that for each tile composing the mapping a normalized total wire length of 7.35 is required.

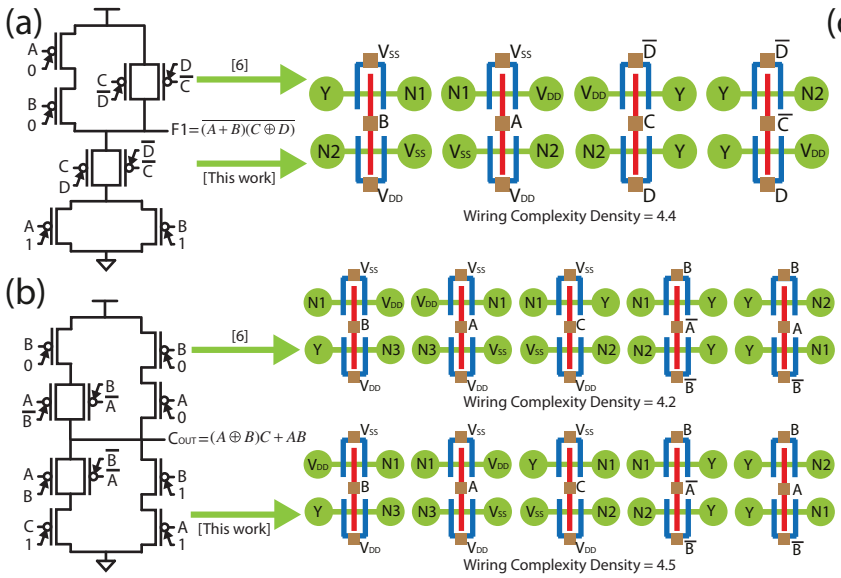
Most gates required only 3 SAT instances for a successful mapping. This corresponds to the minimum number of SAT instances that have to be run (at least one to determine the size of the grid of tiles, one to determine the number of optimization constraints for pins b and one for pins a). This means, as we argued in section III-E, that for most gates, all optimization constraints could be enforced, i.e., all pairs of neighboring pins could be assigned to the same net.

SATSoT mapped successfully gates of up to 32 transistors, most of them in a few tens to a few hundreds of milliseconds. The runtime increases with the number of transistors, but also with the number of nets in a gate. Over the whole libraries, 66% of gates could be mapped in less than 200 ms and the average runtime is 163.3 ms.

3) Discussions: Our methodology is thus capable of achieving compact mappings with good wiring complexity by simply enforcing the sharing of neighboring drain, source an polarity gate pairs: in most cases, all such pairs could be shared. This characteristic makes possible the mapping of most gates with only three SAT instances, enabling short runtimes. The capabilities of the proposed tool enable interesting opportunities in the context of regular fabrics based on emerging devices.

In the present paper, the methodology was presented with Tile G1. However, by slightly modifying the set of constraints, it is possible to adapt it to a broad range of tile geometries. Hence, SATSoT makes possible the evaluation of various tile sizes with regards to their impact on wiring complexity.

In addition, while linked to a synthesis framework such as ABC [20] or MIXSyn [21], SATSoT enables the design of complex circuits by providing an automatic link between logic cells and their efficient implementation onto the SoT. Such an opportunity is of paramount interest, as it allows the designers to study not only the performance of SoT, but also to exploit the new set of logic functions proposed



(c)

Gate	#Trans.	#Tiles	#Hand [8]	W.C.D.	Rt (*ms*)	#SAT
Ambilib						
F03	4	1	N.A.	2.83333	24	3
F13	6	2	N.A.	3.80556	28	3
F15	10	3	N.A.	5.92	40	3
F25	10	3	N.A.	5.86667	38	3
F32	10	4	N.A.	6.02585	350	5
F36	10	4	N.A.	5.86463	306	5
F20	12	3	N.A.	7.66667	54	3
F31	12	3	N.A.	7.625	50	3
F34	12	4	N.A.	7.71429	411	4
F41	12	4	N.A.	6.16071	441	4
CMOS Standard-Cells						
INV_X1	2	1	1	0.5	110	3
NOR2_X1	4	1	1	2.83333	90	3
OAI21_X1	6	2	2	4.05556	89	3
AND3_X1	8	2	2	5.5	34	3
NAND4_X1	8	2	2	4.925	126	3
AOI221_X1	10	3	3	5.51333	50	3
XNOR2_X1	8	2	2	6.00667	188	3
XOR2_X1	8	2	2	5.74	191	3
MUX2_X1	12	3	N.A.	6.33333	111	3
DLH_X1	16	4	N.A.	8.375	288	3
DLL_X2	18	5	N.A.	8.25265	303	3
TBUF_X4	18	5	N.A.	8.67901	1279	3
CLKGATE_X2	22	6	N.A.	9.02112	422	3
BUF_X8	24	6	N.A.	11.1667	282	3
CLKGATEST_X2	26	7	N.A.	10.1943	2185	3
AOI22_X4	32	8	N.A.	11.4883	257	3
NAND4_X4	32	8	N.A.	12.1125	813	3
INV_X32	64	16	N.A.	19	5936	3

Fig. 9. (a) Mapping of F21 gate on Tiles G1. (b) Mapping of carry-out logic functions on Tiles G1. (c) Selected performance results of SATSoT + Minisat on ambilib and a regular CMOS standard cell library mapped on Tiles G2(*#Trans.*: Number of transistors per gate, *#Tiles*: Number of tiles used to map the function, *#Hand*: Number of tiles used to manually map the function [8], *W.C.D.*: Wiring Complexity Density, *#SAT*: Number of running SAT instances)

by controllable-polarity devices [6]. In addition, SATSoT paves the way to library-free mapping approaches, especially promising for the architectural exploration of circuits based on controllable-polarity devices [21]. Finally, the modularity of SATSoT enables the generation of several different mappings for each gate to support a large set of geometrical conformations, simply by modifying the dimensions of the grid, leveraging some extra freedom for any subsequent gate placement step.

V. Conclusion

We presented a methodology using Boolean satisfiability to map netlists of controllable-polarity transistors onto regular SoT fabric. We implemented this methodology and ran it on two libraries, one targeting controllable-polarity devices and the other based on conventional CMOS-based gates. Mapping results showed that the presented methodology is able to achieve wiring complexities similar to mappings obtained manually by previous methodologies. It allows us to automatically exploit the sharing of the different terminals of the tiles and achieves a high resource usage, with runtimes from a few tens of milliseconds to around one second for the biggest gates.

Acknowledgments

The authors would like to thank Luca Amarú for the fruitful discussions. This work has been partially supported by the ERC senior grant NanoSys ERC-2009-AdG-246810.

References

[1] V. Subramanian, "Multiple gate field-effect transistors for future CMOS technologies," IETE Tech. Rev., 27(6): 446, 2010.

[2] D. Hisamoto *et al.*, "FinFET-a self-aligned double-gate MOSFET scalable to 20 nm," IEEE Trans. on Electron Devices, 47(12):2320-2325, 2000.

[3] M. De Marchi *et al.*, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs," IEDM Tech. Dig., 2012.

[4] Y.-M. Lin, J. Appenzeller, J. Knoch and P. Avouris, High-Performance Carbon Nanotube Field-Effect Transistor With Tunable Polarities, IEEE Trans. Nanotechnology, 4:481-489, 2005.

[5] N. Harada, K. Yagi, S. Sato and N. Yokoyama, "A polarity-controllable graphene inverter," Applied Physics Letters, 96:12102, 2010.

[6] M.H. Ben Jamaa, K. Mohanram and G. De Micheli, "An Efficient Gate Library for Ambipolar CNTFET Logic," IEEE Trans. on CAD, 30(2):242-255, 2011.

[7] T. Jhaveri, L. Pileggi, V. Rovner and A. J. Strojwas, "Maximization of layout printability/manufacturability by extreme layout regularity," J. Micro/Nanolith. MEMS, 2007.

[8] S. Bobba, M. De Marchi, Y. Leblebici and G. De Micheli, "Physical Synthesis onto Sea-of-Tiles with Double-Gate Silicon Nanowire Transistors," DAC Tech. Dig., 2012.

[9] S. A. Cook, "The complexity of theorem-proving procedures", ACM Sym. on Th. of Comp., 1971.

[10] J. P. Marques-Silva and K. A. Sakallah, "GRASP: a search algorithm for propositional satisfiability," IEEE Trans. on Computers, 48(5):506-521, 1999.

[11] http://www.lri.fr/~simon/?page=glucose

[12] http://minisat.se/Main.html

[13] http://www.cril.univ-artois.fr/~roussel/ppfolio/

[14] S. Devadas, "Optimal layout via Boolean satisfiability," ICCAD Tech. Dig. 1989.Computer-Aided Design, 1989.

[15] G.-J. Nam, K. A. Sakallah and R. A. Rutenbar, "Satisfiability-based layout revisited: detailed routing of complex FPGAs via search-based Boolean SAT," FPGA Tech. Dig., 1999.

[16] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," DAC Tech. Dig., 2012.

[17] http://www.satcompetition.org/

[18] A. Biere, M. Heule, H. van Maaren and T. Walsh "Handbook of Satisfiability," IOS Press, 2009.

[19] http://si2.org/openeda.si2.org/projects/nangatelib

[20] http://www.eecs.berkeley.edu/~alanmi/abc/

[21] L. Amarú, P.-E. Gaillardon and G. De Micheli, "An Efficient Logic Synthesis Methodology for Mixed XOR-AND/OR Dominated Circuits," ASP-DAC Tech. Dig., 2013.