Layout Technique for Double-Gate Silicon Nanowire FETs With an Efficient Sea-of-Tiles Architecture

Shashikanth Bobba, *Member, IEEE*, and Giovanni De Micheli, *Fellow, IEEE*

***Abstract*— As we advance into the era of nanotechnology, semiconductor devices are scaled down to their physical limits, thereby opening up venues for new transistor channel materials based on nanowires and nanotubes. Transistors based on nanowires and nanotubes inherently exhibit ambipolar behavior. While technologists aim to suppress ambipolar behavior of these transistors, new design methodologies are proposed by exploiting the phenomenon of controllable polarity. In this paper, we propose regular layout fabrics, with an emphasis on silicon nanowires (SiNWs) as the candidate technology. A double-gate ambipolar SiNW field-effect transistor operates as p-type or n-type by electrically controlling the polarity of the second gate. We propose layout techniques to address gate-level routing congestion, as every transistor has two gates to route. Novel symbolic layouts, which are technology independent, are proposed for ambipolar circuits. In the second part of this paper, we present an approach for designing an efficient regular layout called sea-of-tiles (SoTs). A logic tile is essentially an array of prefabricated transistor-pairs grouped together. We design four logic tiles, which form the basic building block of the SoT fabric. We run extensive comparisons of mapping standard benchmarks onto the SoT fabric to find the optimum tile. This paper shows that SoT with Tile$_{G2}$ and Tile$_{G1h2}$, on an average, outperforms the one with Tile$_{G1}$ by 16% and 14% in area utilization, respectively.**

***Index Terms*— Ambipolar, double gate (DG), layout, nanowire, physical design, tiles.**

I. Introduction

FOLLOWING the trend to 1-D structures, silicon nanowire field-effect transistors (SiNWFETs) are a promising extension to the tri-gate FinFETs [1]. The superior performance of these 1-D channel devices comes from a high I_{ON}/I_{OFF} ratio, due to the gate-all-around structure, which improves the electrostatic control of the channel, thereby reducing the leakage current of the device. The advantage of SiNWFETs over other 1-D devices such as carbon nanotube transistors is that SiNWs can be fabricated with a top–down silicon process [2]. In addition, SiNWs can be built in vertical stacks, thereby giving highly dense array of nanowire transistors [3]. Fig. 1(a) and (b) shows a possible extension of a FinFET to SiNWFET device structure with SiNWs suspended between source and drain pillars. In addition, SiNWFETs exhibit enhanced electrostatics properties, such as polarity control, which are electrically hard to achieve in planar- and FinFETs.

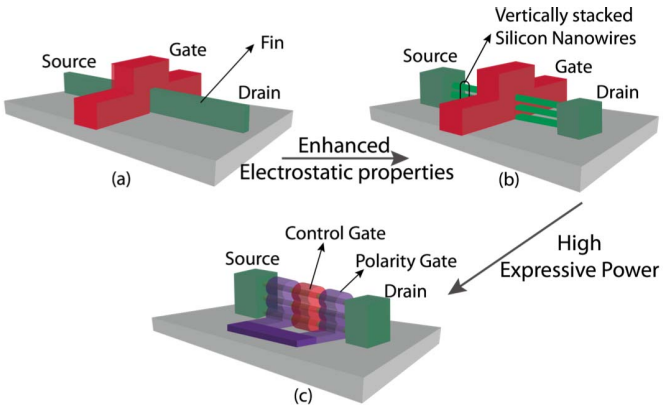

Fig. 1. (a) FinFET providing increase in controllable channel area between the source and drain regions. (b) Vertically-stacked SiNWFET with multiple parallel nanowire channels, each with gate-all-around control. (c) DG-SiNWFET with control and PGs.

Our methodology takes advantage of the electrostatics of these devices, which can be fabricated to be ambipolar, i.e., to exhibit both n-type and p-type characteristics. By engineering the source and drain contacts and by constructing independent double gate (DG) structures, the device polarity can be electrostatically forced to either n-type or p-type by polarizing one of the two gates. Fig. 1(c) shows a DG-SiNWFET device structure with control gate (CG) and polarity gate (PG). The in-field polarizability of these devices enables the development of new logic architectures, which are intrinsically not implementable in CMOS in a compact form [4]. However, the routing complexity at the device level increases due to the presence of an extra gate, the PG [5].

Typical CMOS layout techniques involve transistors with a single gate. In the traditional approach for CMOS, compact layouts are realized by optimal transistor chaining of p-type and n-type transistors [6]–[8]. However, in the case of ambipolar gates, the polarity of the transistor (p-type or n-type) changes with the input signals. Motivated by these observations, we propose compact layout techniques for DG-SiNWFET. To facilitate this, we propose novel

Manuscript received November 14, 2012; revised June 23, 2013; accepted August 12, 2013. Date of publication October 8, 2014; date of current version September 23, 2015. This work was supported by the European Research Council under Grant ERC-2009-AdG-246810.

The authors are with the Integrated Systems Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: shashikanth.bobba@epfl.ch; giovanni.demicheli@epfl.ch).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2358884

symbolic layouts for ambipolar logic with dumbell–stick diagrams (DSDs).

Layout regularity is one of the key features required to increase the yield of ICs at advanced technology nodes [9]. Hence, design styles based on regular layout fabrics have the advantage of higher yield as they maximize the layout manufacturability. Various regular fabrics have been proposed throughout the evolution of semiconductor industry, where some recent approaches are discussed in [10]–[12]. In gate-array fabric style, a sea of prefabricated transistors is customized to obtain a desired logic gate. The flexibility of building generic logic gates comes at a cost of area as well as routing overhead, thereby increasing the performance gap between application-specific integrated circuit (ASIC) and gate arrays. With the advent via programmable gate arrays [11] and logic bricks [12], the performance gap is reduced. On the other hand, strict design rules, at 22-nm technology node and beyond, has led to cell layouts with arrays of gates with a constant gate pitch, which resemble a sea-of-gates layout style.

In this paper, we design an efficient regular layout brick (called as tile), which forms the basic building block for sea-of-tiles (SoT) design methodology. The basic tile for SoT is optimized for area and regularity. Technology mapping, with logic synthesis tools, on various tiles helped us in choosing an efficient tile for realizing SoT. With the optimal tile as a basic building block for a SoT fabric, we demonstrate mapping any 3-input NPN-equivalent function [13], [14] as well as other ambipolar logic circuits.

The main contributions of this paper are as follows.

1) A brief discussion on realizing various types of Boolean functions realized with controllable-polarity transistors, which sets the foundation of ambipolar logic circuits for digital IC design.
2) We address the gate-level routing issues of ambipolar circuits, with emphasis on DG-SiNWFETs. We propose compact layout techniques for complex gates with embedded XOR functions. To facilitate this, we propose novel symbolic layouts for ambipolar logic with DSDs.
3) We design an efficient regular layout brick (logic tile), which forms the basic building block for a SoT design methodology. We present a few case studies by mapping various logic functions onto an optimized SoT fabric.

The preliminary version of this paper, presented in [15], has been extended with a detail study on realizing various Boolean functions with ambipolar logic. In this paper, we also present a layout synthesis procedure to generate an efficient layout of a complex logic function with embedded XOR operation.

The remainder of this paper is organized as follows. Section II provides a background on DG-SINWFET technology and device operation. Section III discusses ambipolar logic circuits by realizing various Boolean functions with controllable-polarity transistors. Section IV introduces novel layout techniques for ambipolar circuits by mitigating gate-level routing overhead caused by an extra gate for every transistor. In Section V, we propose an efficient layout fabric

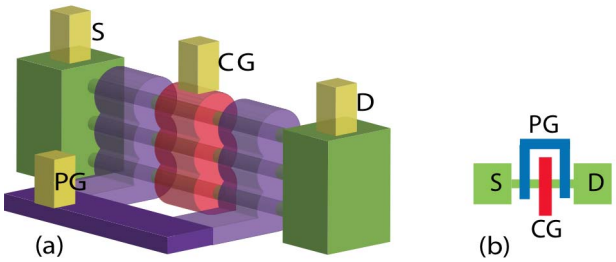

Fig. 2. Conceptual structure of the ambipolar DG-SiNWFET. (a) 3-D view of the device. (b) Top view of the device showing one stack of nanowires forming the channel.

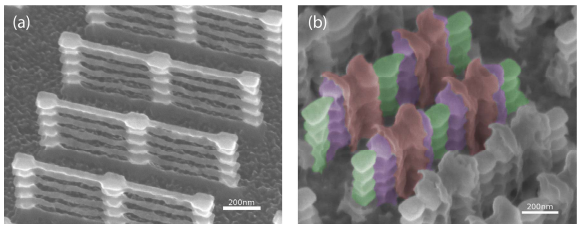

Fig. 3. SEM images of a DG vertically stacked SiNWFET. (a) Before the gate patterning. (b) After the gate patterning: CG (red), PG (violet), and active area (green).

called as tile, which forms the basic building block for SoT design methodology. Finally, the conclusion is drawn in Section VI.

II. Background and Motivation

This section surveys previous works related to ambipolar technologies with a main focus on DG-SiNWFET. It also summarizes various new design techniques, which leverage ambipolarity at the circuit level.

A. Ambipolar DG-SiNWFET Technology

Various new technologies show an inherent behavior toward controllable polarity, including SiNWFETs [16], carbon nanotube FET [17], and graphene nanoribbons [18]. In this paper, we focus on SiNWFETs to illustrate the layout technique for ambipolar logic circuits. The advantage of SiNWFETs over other 1-D devices such as carbon nanotube transistors is that SiNWs can be fabricated with a conventional silicon process as an extension to traditional CMOS technology [2]. In addition, SiNWs can be built in vertical stacks, thereby giving dense arrays of nanowire transistors [3].

Fig. 2 shows a DG-SiNWFET device structure with SiNWs suspended between source and drain pillars. This SiNW is divided into three sections, which are in turn polarized by two gate-all-around gate regions. The center gate region works as in a conventional MOSFET, switching conduction in the device channel by means of a potential barrier. The side regions are instead polarized by a PG, which controls the Schottky barrier thicknesses at the source/drain (S/D) junctions and selects the majority carrier type, thus forcing the device to be either n-type or p-type.

A single electron microscope (SEM) image of an array of vertically stacked SiNWs, suspended between pillars, before patterning the gates, is shown in Fig. 3(a). Fig. 3(b) shows the DG-SiNWFET after patterning the control and PGs [5].

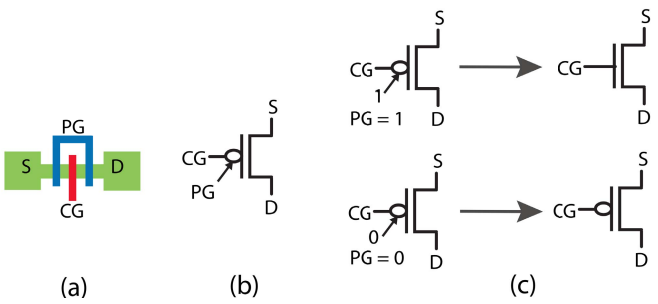

Fig. 4. DG-SiNWFET. (a) Layout (top view). (b) Symbol of an ambipolar FET. (c) Configuration as n-type and p-type by setting the PG.

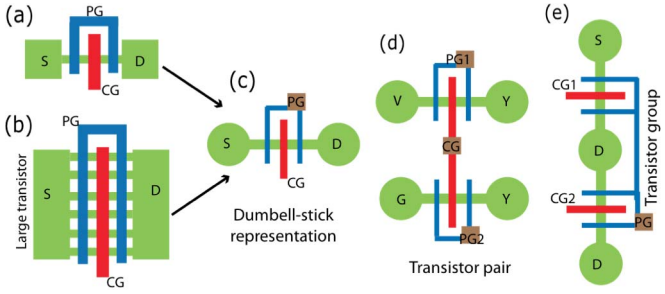

Fig. 5. (a) Top view of the DG-SiNWFET shown in Fig. 1. (b) Large transistor. (c) Equivalent DSD. (d) DSD of an inverter with a transistor pair. (e) Grouping transistor with similar PGs.

B. Device Operation

A fabrication technique to manufacture programmable DG-SiNWFETs has been proposed in [5]. Fig. 4(a) and (b) shows the top view of the DG-SiNWFET and its corresponding symbol. As the name suggests, the device has two gates CG and PG. The CG is similar to the regular gate of a MOSFET, which turns the device ON or OFF. On the other hand, PG sets the majority carriers of the device channel to either p-type or n-type. As shown in Fig. 4(c), if the PG is set to high (logic 1), the device behaves as a n-type transistor, and by setting the PG to low (logic 0), we obtain a p-type transistor.

Though we focus on DG-SiNWFETs, the proposed design methodology holds relevant for other ambipolar FETs (CNFETs [17] and GNRFETs [18]) with two independent gates for in-field programmability.

C. Previous Design Approaches

New design methodologies are proposed for exploiting the controllable polarity, unique to DG devices, which leads to a very compact realization of XOR function [19]–[21]. In [19], a reconfigurable logic gate that maps eight different 2-input logic functions in dynamic logic was presented. In [20], a library of static ambipolar gates based on generalized NOR–NAND-AOIs is proposed that efficiently implements XOR-based functions. Various novel reconfigurable blocks with embedded XOR blocks have been proposed that leverage upon embedded XOR functionality [22]. Zukoski *et al.* [21] proposed universal logic modules that leverage ambipolar transistors. In this paper, we abstract the physical design issues that are common to all the new design methodologies comprising of DG ambipolar transistors. We also propose a procedure for constructing the symbolic layout of ambipolar logic circuits.

III. Ambipolar Logic Circuits and Symbolic Layouts

In this section, we first propose novel symbolic-layouts for controllable-polarity logic gates called DSDs. In the second part, we discuss various logic implementations with ambipolar DG transistors.

A. Terminology

A controllable polarity transistor, αt, is denoted as a quadruple (D, CG, PG, and S) signals that αt connects to, respectively. The voltage signal applied to the PG determines the type (p-type or n-type) of transistor. A transistor, αt, operates as a p-type device (αt_p) by connecting the PG to 0, and an n-type device (αt_n) by connecting the PG to 1.

B. Symbolic Layouts for Ambipolar Logic: DSDs

Similar to the CMOS stick diagrams [23], DSDs denote ambipolar devices (in our case DG-SiNWFET) with a simplified layout abstraction to study the cell-routing complexity. Fig. 5(a) shows the top view of a simple DG-SiNWFET (Fig. 2). As for FinFETs, a large transistor is obtained by increasing the number of nanowire-stacks (fins in the case of FinFET) in parallel, as shown in Fig. 5(b). In Fig. 5(c), we show the dumbell–stick representation of the transistor, with suspended silicon nanowires between the source and drain contacts forming the basic dumbell, and the CG and the PG constituting the sticks. It has to be noted that DSDs do not consider the size of the transistor, but just the topology of the interconnect.

Transistor pairing, shown in Fig. 5(d), is an important transistor placement technique used for layout area reduction. By transistor-pairing, CGs of two transistors (connected to the same signal) are vertically aligned by a single stick segment to minimize the routing complexity as well as to ensure more layout regularity. Two transistors, αt_1 and αt_2, are paired together if their CGs are connected to the same signal, i.e., $\mathrm{CG}(\alpha t_1) = \mathrm{CG}(\alpha t_2)$.

In Fig. 5(e), we show transistor grouping. Two transistors, αt_1 and αt_2, belong to the same group if their PGs are connected to the same voltage, i.e., $\mathrm{PG}(\alpha t_1) = (\alpha t_2)$. Hence by transistor-grouping, transistors with similar PGs are grouped together. Transistor grouping is unique to ambipolar DG devices. In the following section, we show the importance of grouping transistors for minimizing the routing overhead introduced by PGs.

C. Unate, Binate, and Mixed Boolean Functions [24]

A function $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$ is positive unate in x_i if for all x_j, $j \neq i$

$$f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \geq f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).$$

TABLE I
EXAMPLE OF UNATE, BINATE, AND MIXED FUNCTIONS

Boolean function	Type
INV, NAND2…	unate (negative)
AND2, OR3…	unate (positive)
XOR (a, b)	binate
$sa + \bar{s}\,\bar{b}$	mixed (binate + postive unate + negative unate)
$sa + \bar{s}b$	mixed (binate + positive unate)
$(sa + \bar{s}b)'$	mixed (binate + negative unate)
$\overline{a\,(b\ \mathrm{XOR}\ c)}$	mixed (XNUmixed)

Similarly, it is negative unate in x_i if, for all x_j, $j \neq i$

$$f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \geq f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n).$$

A function f is binate in variable x_i if it is neither positive nor negative unate on variable x_i. A function is (positive/negative) unate if it is either positive or negative unate for all x_i, where $i \in [1, n]$. Similarly, a function is binate if it is binate for all the variables. A function is mixed, if it contains both binate and unate variables. Table I gives few examples of unate, binate, and mixed functions.

There are various flavors of mixed functions, according to how binate and positive/negative unate variables are combined. We also consider here a subclass of mixed functions that is common in design libraries. We call XNUmixed those functions that are conjunctions/disjunction of XOR/XNOR with negative unate functions. De Morgan's law [25] can be used to map into this class those functions that combine positive unate functions with XOR/XNOR.

D. Logic Gates Realized With Controllable-Polarity Transistors

In this section, we describe circuit level implementation of negative unate, positive unate, binate and XNUmixed logic functions realized with controllable-polarity transistors.

1) Negative Unate Functions: Negative unate functions are obtained by biasing the PGs of the pull-up-network (PUN) to Gnd and pull-down-network (PDN) to Vdd. This is similar to complementary CMOS style where the PUN and PDN are comprised of p-FETs and n-FETs, respectively. Fig. 6(a) shows a 2-input NAND gate. Since the ambipolar transistors are configurable, just by swapping the Vdd and Gnd terminals, along with the connection to the PGs, of the NAND schematic [Fig. 6(a)], we generate a NOR function, as shown in Fig. 6(b). This technique applies to all the negative unate function.

2) Positive Unate Functions: In the case of positive unate functions, various design approaches are considered. Fig. 6(c) shows an implementation of a 2-input OR gate from the same schematic of a NAND gate. By interchanging the voltage applied to the PGs in the PUN and PDN, we obtain n-type and p-type transistors in the PUN and PDN, respectively. Though this gives a straight forward implementation of positive unate logic, we have to consider the degraded output signal (e.g., $(A + B)_d$). By adding a buffer at the output we can realize full swing at the output. On the other hand, a positive unate function can be obtained by inverting the output of an equivalent negative unate function. An example of a 2-input AND gate is shown in Fig. 6(d). In addition, a positive unate function can be obtained by applying De Morgan's law to the function, as shown in Fig. 6(e). Since we prefer not to use a configuration that degrades output signals and requires buffer [see Fig. 6(c)], a rule of thumb to implement positive unate functions is by biasing the PGs of the PUN and PDN to Gnd and Vdd, respectively [Fig. 6(d) and (e)]. Between the two configurations of Fig. 6(d) and (e), we can observe that the implementation in Fig. 6(d) is better as it requires fewer numbers of inverters (e.g., input inversion has to be accounted for).

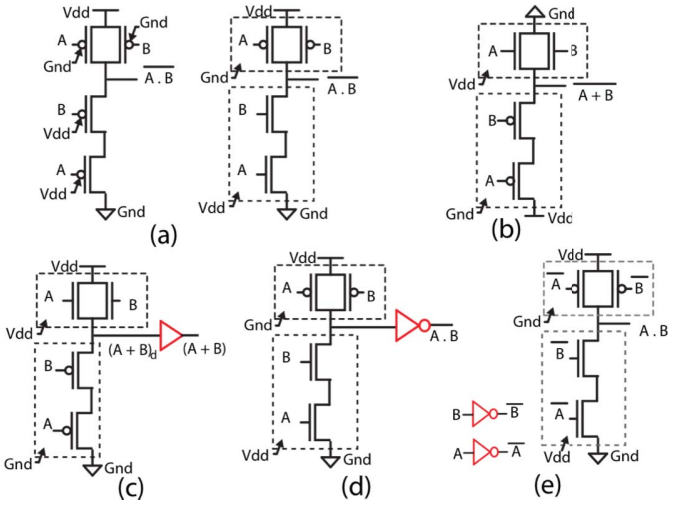

Fig. 6. Unate logic function. (a) NAND gate. (b) NOR gate implementation by swapping the Vdd and Gnd of a NAND gate (a). (c) OR gate implementation by interchanging the voltage of the PGs in the PUN and PDN. (d) AND (positive unate) gate implemented with NAND (negative unate) gate followed by inverter. (e) AND gate implemented by applying De Morgan's rule.

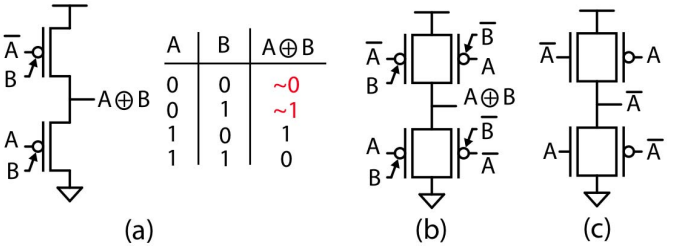

Fig. 7. 2-input logic gates. (a) NAND gate with PGs connected to Vdd and Gnd. (b) XOR gate with PGs connected to input signals (B or $\bar{B}$). (c) XOR gate when the input signal B is assigned to logic 1.

3) Binate Functions: The DG transistors are efficient in implementing binate functions. An example of a 2-input XOR gate with only two ambipolar transistors is shown in Fig. 7(a). When compared with unate logic style, we notice that the PGs are connected to the input logic signals (e.g., logic signal B in Fig. 7). From the truth table shown in Fig. 7(a), we observe that output is degraded when the ambipolar transistor is configured to be p-type in the PDN and n-type in the PUN. The degraded output signal can be recovered by placing a buffer at the output. To obtain full swing at

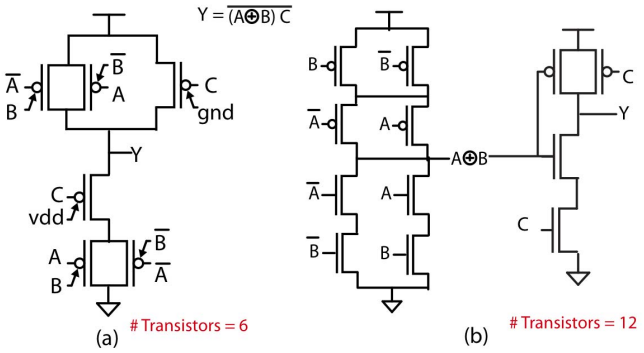

Fig. 8. Partially binate function $Y = \overline{(A \text{ XOR } B)C}$. (a) Ambipolar logic style, where the PGs of the binate logic are connected to logic inputs (B or $\bar{B}$) and PGs of unate variables are connected to Vdd and Gnd. (b) Static CMOS implementation.

the output, an alternative approach using transmission-gates (e.g., two parallel transistors) is proposed in [20], where a 2-input XOR gate can be constructed using only four ambipolar transistors. An example of a 2-input XOR gate is shown in Fig. 7(b), where all the PGs are either connected to logic input B or $\bar{B}$. For any given configuration, the output is either pulled-up (or pulled-down) by both n-type and p-type transistors. Fig. 7(c) shows the case where B is assigned to logic 1. The transmission gates with complemented inputs in the PUN and PDN assure a full swing at the output. When compared with static CMOS implementation of an XOR2 (which needs 12 transistors), the transmission-gate XOR2 with ambipolar transistors needs only eight transistors [transistors shown in Fig. 7(b) along with the two inverters for generating $\bar{A}$ and $\bar{B}$].

4) Mixed Functions: Within the mixed function class, XNUmixed functions can be effectively laid out. As an example, we show the implementation of function $Y = \overline{(A \text{ XOR } B)C}$ in both static-CMOS logic style [Fig. 8(b)] and ambipolar logic style [Fig. 8(a)]. From the figure, we can observe that the number of transistors is reduced by half with ambipolar logic style when compared with CMOS implementation. We incorporate transistor pairs only for the XOR combination of the logic, where the PGs are biased to input logic signals (B and $\bar{B}$). For the variables, which are negative unate (e.g., logic function Y is negative unate in C), the PGs are connected to the Vdd and Gnd, as shown in Fig. 8(a).

IV. Layout Technique for Ambipolar Logic Gates

One of the caveats of ambipolar design style is the increase in the intracell routing complexity. Since every transistor has two gates to connect to the logic signals, care should be taken to mitigate the gate-level routing complexity. In this section, we propose a novel layout technique for ambipolar design style. In addition to transistor pairing, we also leverage on transistor grouping, thereby obtaining layouts that are compact, regular, and easy to route. We start with simple examples of 2-input logic gates and then propose a generic procedure for arbitrary complex XNUmixed gates.

A. Layout Techniques for a 2-Input Unate and Binate Functions

From Section III, we have seen that negative unate logic gates (e.g., NAND, NOR, INV,...) can be obtained by biasing the PGs of the PUN to Gnd and PDN to Vdd. Hence, all the transistors in the PUN (and PDN) can be grouped together (i.e., PGs of the stacked transistors are connected together), thereby forming one PG for each PUN and PDN. With fixed biasing for the PGs, CMOS layout techniques with optimal transistor chaining [7], [8] can be employed to obtain area-efficient layout. The transistors are placed in two parallel rows where all transistors in the PUN are in one row, while all the transistors in the PDN are in the other. The main objective is to place transistors in such a way that the gate signals are aligned and D/S regions of adjacent transistors are abutted. Fig. 9(a) shows an example of a 2-input NAND gate with an Euler path approach [6]. From the Euler path, the optimal transistor alignment chain is obtained.

On the other hand, for positive unate logic gates (e.g., AND, OR, BUF,...), one of the techniques shown in Fig. 6(c)–(e) can be employed. In all the three cases, we can observe that the transistors in the PUN and PDN can be grouped together and tied to either Vdd or Gnd. The layout technique for unate logic gates is similar to CMOS style.

The main application of ambipolar devices is in implementing binate logic functions. From Section III, we have seen that a 2-input XOR gate can be constructed using only four transistors. Fig. 9(b) shows an example of a 2-input XOR gate along with the two possible dumbell–stick representations. In case-1, we illustrate CMOS style layout where the transistors in PUN and PDN network are paired by realizing Euler paths in the respective networks. It has to be noted that the PGs in the PUN (and PDN) cannot be grouped, unlike in the case of unate logic gates. Since the adjacent transistors cannot be grouped, extra routing effort is needed to connect PGs together [case-1 of Fig. 9(b)]. An efficient implementation is shown in the case-2 of Fig. 9(b), where polarity gates are grouped together irrespective of the transistor being a part of PUN or PDN. The circuit is no more seen as PUN and PDN, but partitioned based on the signals assigned to the PGs. From the DSD, we can observe that the PUN and PDN are placed next to each other. Unlike CMOS, DG-SiNWFET technology does not impose any process challenges (which lead to design rules) when placing p-type next to n-type transistors.

B. Layout Techniques for XNUmixed Function

Several novel circuit designs and architectures have been proposed that leverage upon ambipolar logic with embedded XOR functionality [20]–[22]. De Marchi *et al.* [22] have presented the idea of regular logic fabrics and evaluated various complex gates (combination of AND–XOR–OR–INV) based on the number of subfunctions each gate can implement. A key observation is that 2-input XOR/XNOR gates form the main building block of most logic cells, especially used in data-path design and within arithmetic building blocks. Recall that XNUmixed functions are conjunctions/disjunction of XOR/XNOR with negative unate functions.

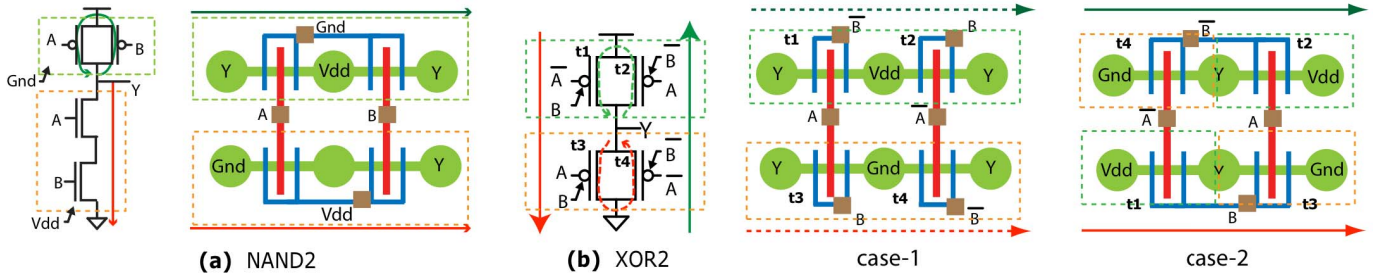

Fig. 9. DSD for 2-input logic gates. (a) NAND gate with the PGs grouped together in the pull-up (pull-down) and connected to GND (VDD). (b) XOR gate—(case-1) conventional approach by placing the transistors in the pull-up (pull-down) together so that they share the diffusion contacts (case-2) efficient layout technique where the transistors are grouped together, irrespective if they are located in the pull-up or pull-down networks, as well as share the same diffusion contacts.

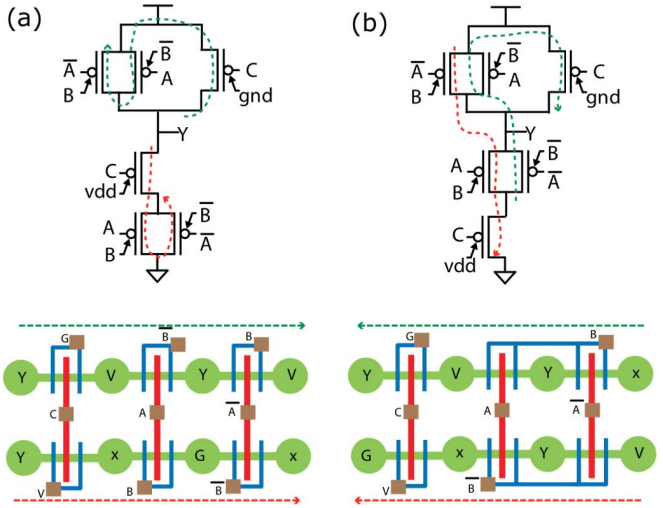

Fig. 10. Transistor ordering for XNUmixed logic function. (a) Binate logic part placed close to Vdd and Gnd terminals. (b) Binate logic part placed close to the output (Y).

From the example of XOR2 [Fig. 9(b)], we observe that efficient layouts can be obtained by placing the transistors together with similar PGs. To facilitate grouping, a specific transistor ordering is needed for XNUmixed logic functions. Fig. 10 shows two different transistor arrangements for the function $Y = \overline{(A \text{ XOR } B)C}$. In Fig. 10(a), the binate logic part (A XOR B) is realized close to the Gnd and Vdd terminals, whereas, in Fig. 10(b) it is realized close to the output (Y) terminal. From the DSDs of the two cases, we can infer that the circuit implementation in Fig. 10(b) is efficient when compared with the one in Fig. 10(a), as it reduces the routing needed by the PGs. As a rule of thumb, in the case of XNUmixed logic gates, placing the binate logic close to the output node leads to efficient layout.

C. Procedure for Generating Layout of a XNUmixed Logic Gate

In this section, we present a generic procedure to generate layout for XNUmixed functions. Our objective is to achieve a regular layout with:

1) transistor pairs aligned to give the least number of breaks in the active regions, which lead to realizing compact layouts (like in CMOS and nMOS logic);
2) the least number of transistor groups, to reduce intracell routing complexity.

It has to be noted that if we refer to the first goal only, procedures for CMOS layout [6], [7] are widely applicable. In particular, the algorithm in [7] gives near-optimum solution with short computing time. In our case, it is important to address both aforementioned goals, and thus we adapt Hwang's algorithm, which we summarize below and exemplify in action. We refer the reader to [7] for details.

Our procedure, to meet both objectives, consists of six steps: 1) reordering; 2) grouping; 3) pairing; 4) generating unate- and binate-bipartite graphs; 5) chaining; and 6) DSD construction. Input to the procedure is a XNUmixed circuit schematic with a complementary logic style (i.e., equal number of transistors in the pull-up and pull-down network with dual topology graphs).

The first step is transistor reordering, with the binate inputs placed close to the output node as explained in Fig. 10. By means of transistor grouping, various subgraphs are formed by clustering the transistors sharing similar PGs. We model the circuit schematic as a list of graphs $G = \{G_{\text{PG}-1}, G_{\text{PG}-2}, \ldots G_{\text{PG}-i}, \ldots\}$, where $G_{\text{PG}-i} = (V, E)$, in which V represent the nodes (S/D contacts of the transistors) and E represent the edges (CG of at) of all the transistors whose PG is connected to i. Applying transistor grouping to circuit schematic shown in Fig. 11(a), we obtain $G = \{G_{\text{PG}-v}, G_{\text{PG}-g}, G_{\text{PG}-B}, G_{\text{PG}-\bar{B}}\}$ for the four transistor groups with PGs connected to Vdd, Gnd, B, and $\bar{B}$, respectively. As an example, we list the graph related to the transistors whose PGs are connected to B, $G_{\text{PG}-B} = \{[A, \bar{A}], [(a3, a4), (b2, b1)]\}$.

Transistor pairing is performed next. In this step transistors with similar CGs are paired together. For complementary logic style, each pair consists of a transistor in the PUN and PDN. This step ensures the control gates are well aligned with minimum routing resources.

We differentiate from Hwang's approach by generating separate bipartite graphs for the unate and binate parts of the function. The unate and binate logic part of the circuits can be determined from the transistor-grouping step. We represent the possible abutments between the dual graphs as a bipartite graph G_x. An unate–bipartite graph (G_u)

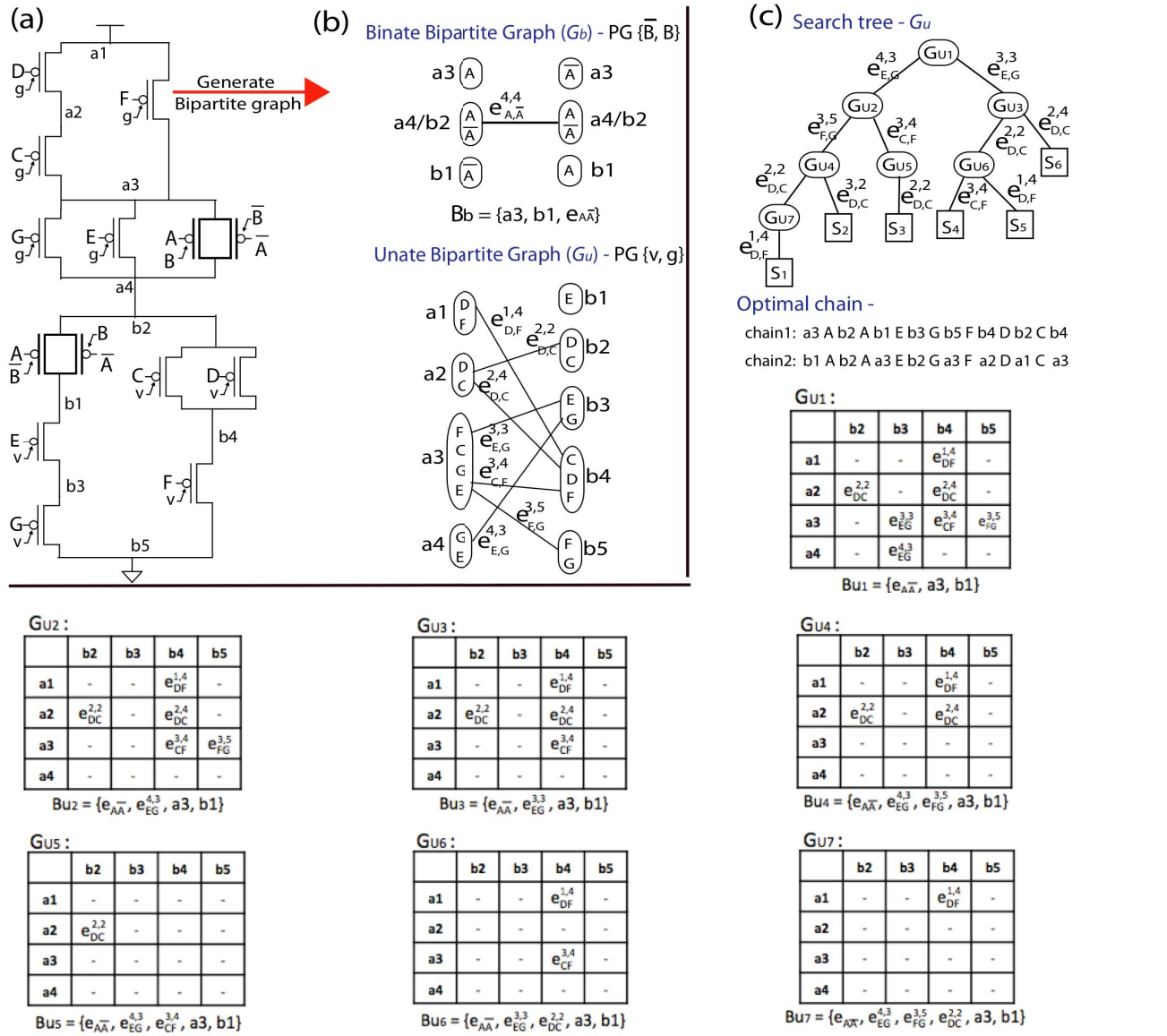

Fig. 11. Logic-to-layout procedure. (a) Complex logic function. (b) Separate bipartite graph representation for binate and unate part of the logic. (c) Search tree of unate-bipartite graph in (b) along with the matrix representation of the nodes of the tree.

corresponds to the dual subgraphs $G_{\mathrm{PG}-v}$ and $G_{\mathrm{PG}-g}$, whereas, a binate–bipartite graphs (G_b) corresponds to the dual subgraphs $G_{\mathrm{PG}-B}$, and $G_{\mathrm{PG}-(\bar{B})}$. In the bipartite graph, nodes with only one transistor contribute to the list of essential abutments (e.g., nodes $a3$ and $b1$ of the graphs G_u and G_b). The main objective of this step is to find a unique transistor chain for the PUN and PDN with minimum number of breaks in the adjacent PGs and the diffusion area.

A pseudocode description of the proposed procedure is shown in Fig. 12. In the algorithm, B_b and B_u represents the set of essential abutments of the bipartite graphs G_b and G_u, respectively. Since the XOR2 part of the logic constitutes mainly for G_b, finding the essential abutments (B_b) is simple, as shown in Fig. 11(b). Once we have the set of essential abutments from the binate logic part of the circuits, we continue to find the essential edges from the remaining part of the circuit. Optimal transistor chaining is obtained by a depth-first search on G_u, while B_u is set to B_b. Fig. 11(c) shows how the procedure works on the example circuit. The search process starts from the root, where $G_{u1} = G_u$ and $B_1 = \{a3, b1, e_{(\bar{A})A}\}$. From G_u [Fig. 11(b)], we see $b1$ as

PROCEDURE *ambipolar_logic_to_layout()*

```
1.  re-ordering();
2.  grouping();
3.  paring();
4.  Gb = binate_bipartite_graph();
5.  Gu = unate_bipartite_graph();
6.  Bb = NULL;
7.  Eabu = essential_binate_abutments(Gb, Bb);
8.  Bu = Eabu;
9.  opt_chain = Chaining(Gu, Bu)
10. dumbell_stick_diagram(opt_chain)
```

Fig. 12. Procedure for layout generation of ambipolar complex logic gate.

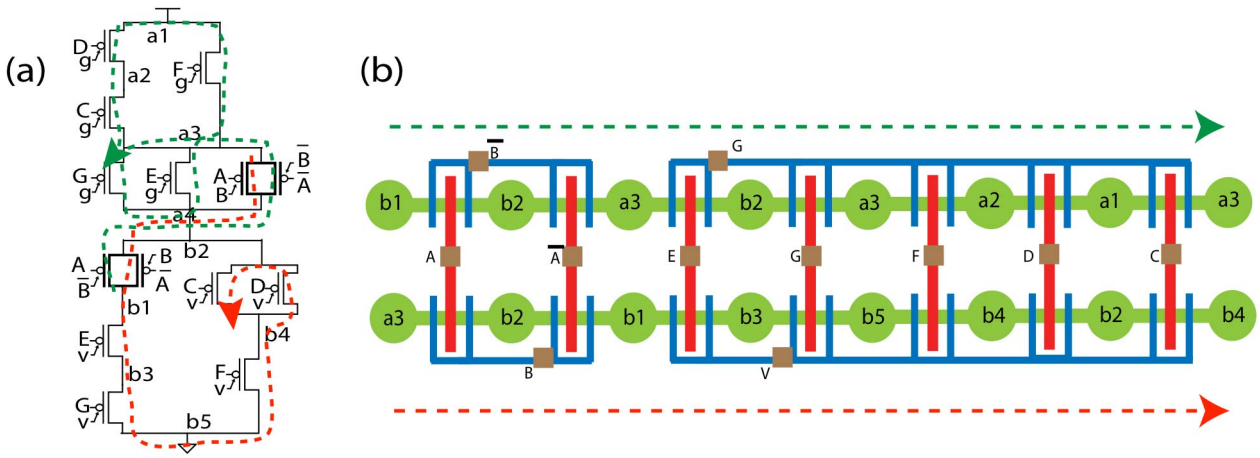

Fig. 13. (a) Graphical representation of transistor chains derived from Fig. 11(c). (b) DSD of the circuit.

an essential edge, hence we form an edge set, which consists of e^{43}_{EG} and its mutually exclusive member, e^{33}_{EG}. Traversing to the left branch of the search tree node, G_{u1}, we add e^{43}_{EG} to B_{u1} to form a new set B_{u2}. Similarly, G_{u2} is derived from G_{u1} by removing edge set corresponding to e^{43}_{EG}. The matrix representations of G_{ui} and B_{ui} at various nodes of the search tree are shown in Fig. 11(c). The leaves of the search represent possible transistor chains (S_i).

A graphical representation of the optimum transistor chain for the example is shown in Fig. 13(a). It can be noticed that, unlike in CMOS layouts, the Euler path spans both the PUN and PDN for obtaining minimum number of breaks in the diffusion region as well as PGs. Fig. 13(b) shows the DSD of the circuit.

D. Relevant Examples

We show here symbolic layouts for DG-SiNWFETs inspired by the literature. Fig. 14(a) shows an efficient reconfigurable logic block ($F1$) with ambipolar transistors, which can implement 12 different subfunctions [22]. Once the DSD is extracted from optimal transistor ordering, the final layout of the circuit is done by considering the sizing of the transistors for uniform delay caused by the transistors in the PUN and PDN.

In Fig. 14(b), we show the carry-out logic implementation with ambipolar logic. An equivalent implementation with conventional static CMOS logic requires 22 transistors, whereas with ambipolar logic we need 16 transistors (10 shown in the figure along with 3 inverters). By applying the layout procedure we obtain four transistor chains, thereby leading to a break in the diffusion region of the DSD.

V. Sea-of-Tiles

Regular layout fabrics have an advantage of higher yield as they maximize the layout manufacturability at advanced technology nodes [9]. Various regular fabrics have been proposed throughout the evolution of semiconductor industry, where some recent approaches are discussed in [10]–[12].

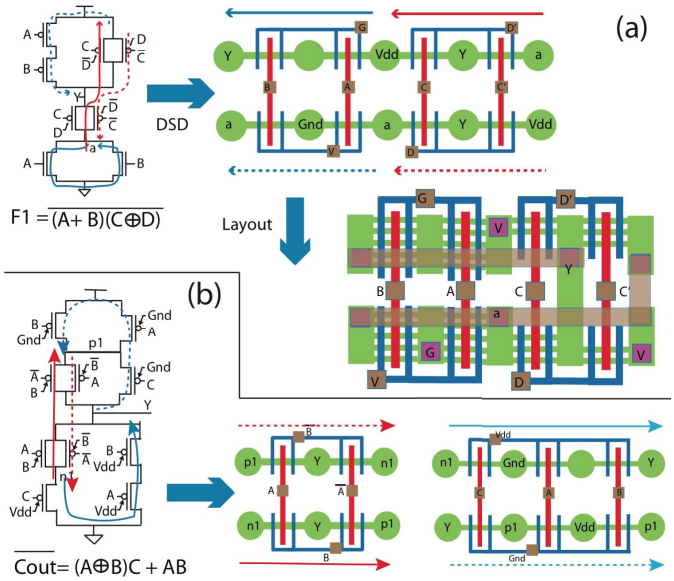

Fig. 14. Examples of complex gates with ambipolar logic. (a) Reconfigurable logic block [3]. (b) Carry out (Cout) function of a full-adder.

With the advent via programmable gate arrays [11] and logic-bricks [12], the performance gap is reduced. On the other hand, strict design rules, at 22-nm technology node and beyond, has led to cell layouts with arrays of gates with a constant gate pitch, which resemble a sea-of-gates layout style. In this paper, we propose a layout fabric architecture called SoTs, for DG-SiNW FETs. In this section, we consider a SoT comprising of a regular set of tiles, and we investigate their effectiveness.

Logic tiles for DG transistors are designed by leveraging both transistor pairing and grouping, thereby leading to an efficient building block for ambipolar circuits. The layout techniques presented in Section IV are employed in the technology-mapping phase, where various logic functions are physically synthesized onto SoT fabric. By technology mapping onto SoT architecture, with different tiles, we investigated an optimal tile by benchmarking various circuits. We show

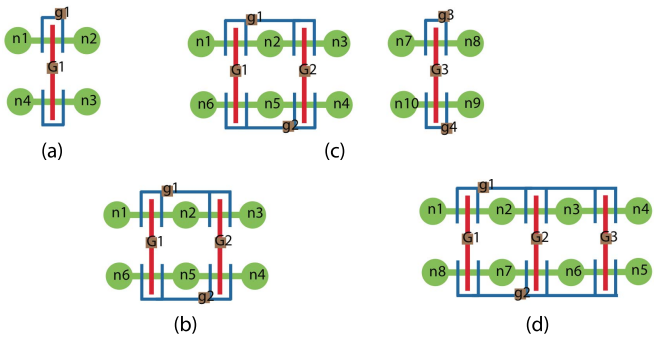

Fig. 15. DSDs of various logic tiles considered for SoTs. (a) Tile$_{G1}$. (b) Tile$_{G2}$. (c) Tile$_{G1h2}$. (d) Tile$_{G3}$.

how tiles form the basic building block for various novel design styles, which are unique to transistors with controllable polarity [21], [22].

A. Logic Tiles are Building Blocks

We define a logic tile as an array of transistors, which are paired and grouped together. By grouping the PGs of the adjacent transistors, we reduce the number of input pins of the tile. In addition, the technology facilitates in realizing these tiles with a high yield as the silicon nanowires are fabricated in groups. In this paper, we limit our study to a maximum of three transistors in series for noise margin reasons. However, the proposed design methodology can be employed to tiles with higher number of series connected transistors.

A Tile$_{Gn}$ is an array of n transistor-pairs grouped together. Fig. 15 shows four tiles that we consider for the SoTs architecture. Any Boolean logic function can be mapped onto an array of tiles. Tile$_{G1}$ [Fig. 15(a)] is the simplest tile with only one pair of transistors. Mapping a generic logic function onto SoT of Tile$_{G1}$, leads to larger layouts with a large number of diffusion breaks and increases in the number of interconnections per tile. Tile$_{G2}$ and Tile$_{G3}$ include two and three transistor pairs, respectively, grouped together. In the example of carry-out logic gate of a full-adder (Fig. 14), Tile$_{G2}$ and Tile$_{G3}$ are employed to realize the gate. Similarly in the case of NAND and XOR (Fig. 9) Tile$_{G2}$ forms the basic building block. A hybrid tile Tile$_{G1h2}$ is a combination of Tile$_{G1}$ and Tile$_{G2}$, whose PGs are not connected. This gives the flexibility of utilizing a part of a tile, when remained unmapped, by functions with low area utilization. For example, a NAND2 gate when mapped onto a Tile$_{G1h2}$ [Fig. 15(c)] requires only the segment of a tile with gates $G1$ and $G2$. The unmapped part of the tile with gate $G3$ can be employed either to map an inverter or to increase the drive strength of the gate.

The Tile$_{G2}$, shown in Fig. 15(b), can be configured to various logic functions by connecting the nodes ($n1$–$n6$) and gates ($g1$, $g2$, $G1$, and $G2$) to appropriate inputs. Table II lists various logic functions that can be realized with a single Tile$_{G2}$. However, any complex logic functions can be obtained by considering an array of Tile$_{G2}$. In Table III, we report various logic gates that can be configured with the four tiles we have considered. The number of tiles required for each gate and their respective area utilization is also presented. It has to be noted that we also consider extra logic needed for generating inverted inputs. For example the 2-input XOR gate, shown in case-2 of Fig. 9(b), is realized with one Tile$_{G2}$ as we assume the availability of complimented input signals. In our technology mapping, we assume single-rail logic; hence we need to generate the complimented signals when needed. In the case of XOR gate, we take an extra Tile$_{G2}$ for generating the two negated input signals ($\bar{A}$ and $\bar{B}$). In the case of hybrid tile, Tile$_{G1h2}$, the number of tiles reported ($\#N$) is a noninteger value as we employ the unmapped part of the tile to a different logic. For instance in the case of mapping an inverter (INV) onto a Tile$_{G1h2}$, only the part of tile with a single transistor pair if employed thereby yielding 100% active area utilization. The unmapped part of this tile can be employed to realize other logic functions (e.g., NAND2 or NOR2).

TABLE II
VARIOUS LOGIC GATES THAT CAN BE REALIZED BY CONFIGURING THE TILE$_{G2}$

Logic	n1	n2	n3	n4	n5	n6	G1	G2	g1	g2
XOR2	Gnd	Out	Vdd	Gnd	Out	Vdd	A	A′	B′	B
XNOR2	Gnd	Out	Vdd	Gnd	Out	Vdd	A	A′	B	B′
NAND2	Out	Vdd	Out	Out	-	Gnd	A	B	Gnd	Vdd
NOR2	Vdd	-	Out	Out	Gnd	Out	A	B	Gnd	Vdd
INV	Vdd	Out	Vdd	Gnd	Out	Gnd	A	A	Gnd	Vdd
BUF	O1	Vdd	Out	Out	Gnd	O1	A	O1	Gnd	Vdd

TABLE III
VARIOUS LOGIC GATES THAT CAN BE MAPPED BY CONFIGURING THE CONTACTS AND THE INPUT SIGNALS OF THE FOUR TILES (#N—NUMBER OF TILES, AND #UF— UTILIZATION FACTOR)

Gates	Tile$_{G1}$		Tile$_{G2}$		Tile$_{G1h2}$		Tile$_{G3}$	
	#N	#UF	#N	#UF	#N	#UF	#N	#UF
AND2	3	0.6	2	0.6	1	0.75	1	1
AND3	4	0.57	2	0.8	1.38	0.67	2	0.57
AOI21	3	0.6	2	0.6	1	0.75	1	1
AOI221	5	0.56	3	0.625	1.62	0.71	2	0.71
AOI222	6	0.54	3	0.75	2	0.67	2	0.86
AOI22	4	0.57	2	0.8	1.38	0.67	2	0.57
AOI321	6	0.54	3	0.75	2	0.67	2	0.86
BUF	2	0.66	1	1	0.62	1	1	0.67
INV	1	0.66	1	1	0.38	1	1	0.67
NAND2	2	0.66	1	1	0.62	1	1	0.67
NAND3	3	0.6	2	0.6	1	0.75	1	1
NAND4	4	0.57	2	0.8	1.38	0.67	2	0.57
NOR2	2	0.66	1	1	0.62	1	1	0.67
NOR3	3	0.6	2	0.6	1	0.75	1	1
NOR4	4	0.57	2	0.8	1.38	0.67	2	0.57
OAI21	3	0.6	2	0.6	1	0.75	1	1
OAI22	4	0.57	2	0.8	1.38	0.67	2	0.57
OR2	3	0.6	2	0.6	1	0.75	1	1
OR3	4	0.57	2	0.8	1.38	0.67	2	0.57
XNOR2	8	0.57	2	0.8	1.38	0.67	2	0.57
XNOR3	9	0.56	3	0.625	1.62	0.71	2	0.71
XOR2	8	0.57	2	0.8	1.38	0.67	2	0.57
XOR3	9	0.56	3	0.625	1.62	0.71	2	0.71

B. Optimal Tiles With Respect to Active Area

In this paper, we compare four tiles for an efficient implementation of the SoT architecture. Our main objective is to find the best tile, which gives highest area utilization for various benchmarks. Though the techniques presented in this

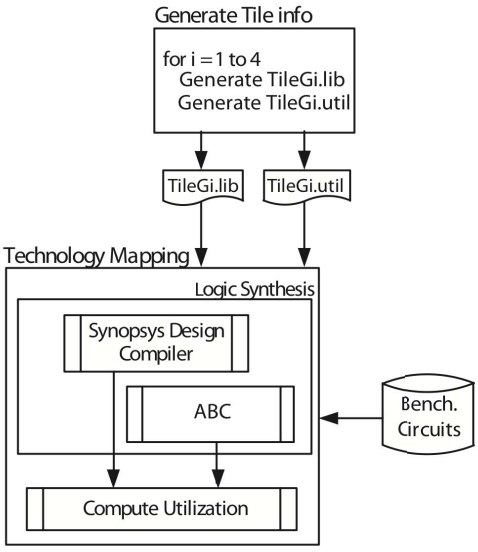

Fig. 16. Design flow for finding the best tile for SoT.

paper are linked to the ambipolar SiNWFETs, the concepts can be extended to all the technologies contending for ambipolar logic circuits with DGs transistors.

Fig. 16 shows our design flow. As a first step, for every tile (Tile$_{Gi}$) we generate a list of logic gates that can be mapped onto it (TileGi.lib) and their respective utilization factor (TileGi.util). Utilization factor takes only the active area into account. For example NAND2 when mapped onto a Tile$_{G1}$ has a utilization factor of 0.66, whereas when mapped onto a Tile$_{G2}$ it has a utilization factor of 1. It has to be noted that the number of logic gates that can be mapped to different tiles vary. For technology mapping, we used Synopsys design compiler [26] and ABC [27] synthesis tools to benchmark various circuits.

Table IV summarizes the results of various benchmark circuits after technology mapping. We report total area utilization for each benchmark when mapped onto four different tiles (Tile$_{G1}$, Tile$_{G2}$, Tile$_{G1h2}$, and Tile$_{G3}$). Technology mapping only uses the cells that are associated with each tile (shown in Table III). Both the synthesis tools were run with different delay constraints. Area utilization for a benchmark circuit is calculated from the total count of each cell and their respective utilization factors.

Examining the results for the four logic tiles, on an average across various benchmark circuits, we see that SoT with tiles Tile$_{G1h2}$ (and, Tile$_{G2}$) have a higher area efficiency, 16% (4%) and 10% (8%), when compared with SoT with Tile$_{G1}$ and Tile$_{G3}$, respectively. Though Tile$_{G3}$ and Tile$_{G1h2}$ have the same number of transistors per tile, the hybrid tile outperforms Tile$_{G3}$ with 10% improvement in area efficiency.

C. Case Study: Mapping Various Blocks Onto SoT

Embedded XOR functionality is one of the key features of ambipolar logic gates. With a transmission-gate transistor structure [20], a 2-input and a 3-input XOR/XNOR gate can be constructed using only four transistors. In Fig. 17, we show how Tile$_{G2}$ can be configured to be XOR2 and XOR3 by connecting the nodes ($n1$, $n3$) and ($n2$, $n4$) to the respective signals shown in the figure. It has to be noted that extra tiles are needed to generate the complemented input signals. In Fig. 18, we show DSDs of both the sum (Sum) and carry-out (Cout) logic of a full-adder, mapped onto a SoT (an array of $n \times n$) with Tile$_{G1h2}$. The layout synthesis procedure, explained in Section IV-C, is applied to obtain the optimal transistor chaining of the Cout logic. Both the Sum and Cout logic blocks are mapped onto three adjacent tiles of the $n \times n$ array. Tile-(i, j) in the figure refers to the location of the tile in ith row and jth column. The Sum, which is a 3-input XOR of inputs A, B, and C, is mapped onto a Tile-(i + 1, j) of the entire array. The unmapped part of the Tile-(i + 1, j) can be employed for realizing either an

TABLE IV
NORMALIZED AREA OF VARIOUS BENCHMARKS WHEN MAPPED ONTO A SoT WITH TILE$_{G1}$, TILE$_{G2}$, TILE$_{G1h2}$, AND TILE$_{G3}$ USING DESIGN COMPILER [26] AND ABC SYNTHESIS [27]

Bench.	Tile$_{G1}$		Tile$_{G2}$		Tile$_{G1h2}$		Tile$_{G3}$	
	DC	ABC	DC	ABC	DC	ABC	DC	ABC
Dalu	1968	2558	1728	2235	1689	2115	1808	2548
Add64	3946	3004	3693	2664	3483	2483	3560	2740
C5315	4072	5404	3465	4791	3422	4477	3984	5088
C7552	4914	5606	4188	5001	4150	4653	4752	5456
i10	5964	6350	5034	5634	4790	5286	5452	6232
C1908	1132	1778	936	1518	942	1469	1116	1692
C3540	2940	3436	2517	3033	2486	2859	2756	3184
C6288	8462	9336	7227	8253	7373	7744	7580	8000
Des	9392	12482	8142	10623	7910	10323	9016	11912
Average	1	1	0.86	0.87	0.85	0.83	0.94	0.94

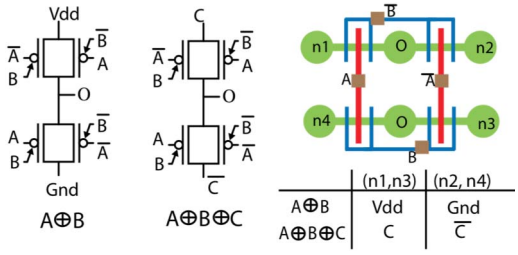

Fig. 17. Schematic of a 2-input and 3-input XOR along with the mapping onto a Tile$_{G2.1}$.

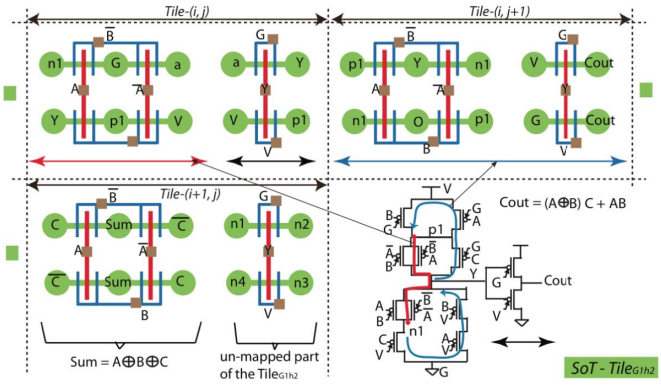

Fig. 18. Full-adder mapped onto a SoTs with the hybrid tile Tile$_{G1h2}$ as the basic building block.

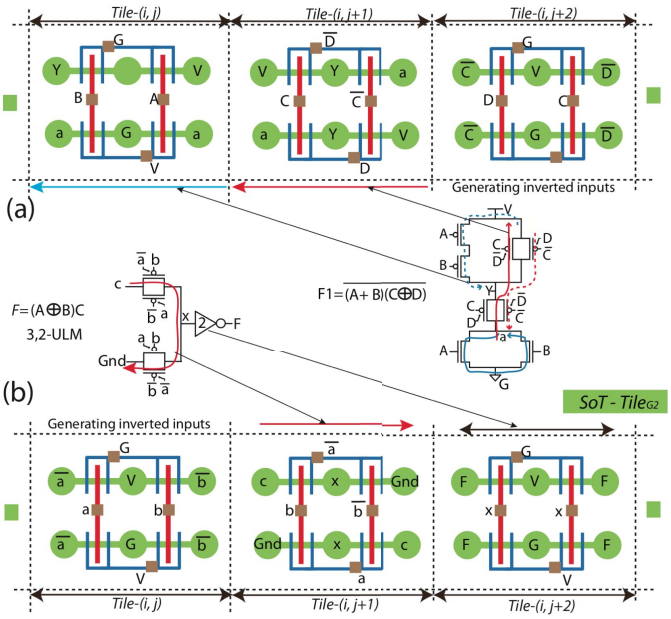

Fig. 19. Reconfigurable fabrics mapped onto SoT with Tile$_{G2}$. (a) Regular computation fabric [3]. (b) Universal logic module (3,2-ULM) [21].

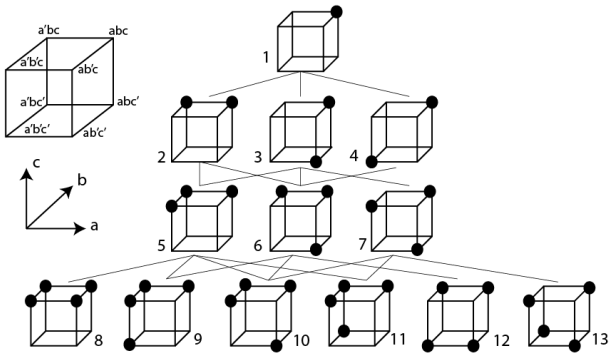

Fig. 20. Matching compatibility graph for 3-input Boolean space.

inverter logic gate or can be a part of the neighboring logic gate. Similarly the Cout is mapped onto two tiles Tile-(i, j) and Tile-(i, $j+1$).

Several novel reconfigurable blocks have been proposed that leverage upon embedded XOR functionality of ambipolar logic. In Fig. 19, we demonstrate how a computational fabric ($F1$) [22] and a universal logic module (3,2-ULM) [21] can be mapped onto a SoT of Tile$_{G2}$. Inverted inputs, for a 2-input XOR functions, are generated with a single tile (Tile-(i, j) for 3,2-ULM and Tile-(i, j+2) for F1).

D. Case Study: Mapping 3-Input Boolean Functions Onto SoT$_{G2}$

In this section, we present physical synthesis of all 3-input Boolean functions onto SoT with Tile$_{G2}$. Fig. 20 shows a matching compatibility graph for 3-input Boolean space [13]. Each vertex V_i in the graph, is annotated with one function F_i, which belongs to the corresponding NPN-equivalence class [14], [28] of V_i. All the functions (F_i), listed in Table V are representative of a NPN-equivalence

TABLE V
NPN-EQUIVALENT FUNCTIONS, OF A 3-INPUT BOOLEAN SPACE, IMPLEMENTED IN STATIC CMOS AND AMBIPOLAR LOGIC STYLES. TYPE OF THE FUNCTION CORRESPONDS TO UNATE (U), BINATE (B), AND MIXED (XNU)

Boolean space	Representative functions	Type	Transistor count	
			Static CMOS	Ambipolar Logic
F_1	abc	U	6	6
F_2	ab	U	4	4
F_3	a(b^c)	XNU	12	6
F_4	abc+a'b'c'	B	12	12
F_5	c(b+a)	U	6	6
F_6	bc+ab'c'	M	10	10
F_7	abc+b'(c^a)	B	18	12
F_8	c	U	2	2
F_9	bc+a'b'	M	8	8
F_{10}	c(b+a') + ab'c'	B	12	12
F_{11}	cb + ca' + a'bc'	M	12	12
F_{12}	(b^c)'	XNU	8	4
F_{13}	a^b^c	XNU	18	10

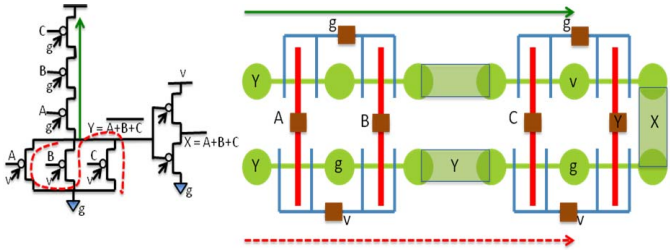

Fig. 21. Layout synthesis for a 3-input OR gate (F_1 equivalent).

class. In other words, all (e.g., 256) 3-input functions can be obtained from the 13 representative functions (in Fig. 20) by input complementation and/or permutation and/or output complementation. The type of the function along with the number of transistors needed for implementing in static CMOS and ambipolar logic styles is listed in Table V. Type of the function refers to it being unate (U), binate (B), mixed (M) and mixed with embedded XOR/XNOR (XNU). We compare the transistor count for realizing the functions with both static CMOS and ambipolar logic implementation. We do not consider the inverters needed for input and output negations, as they are similar for both the logic styles. From the table, we can infer that ambipolar logic style is favorable for 30% of the total NPN-equivalent functions, which have embedded XOR/XNOR (F_3, F_7, F_{12}, and F_{13}).

Layout synthesis technique, presented in Section IV, is applied to the functions listed in Table V for mapping them onto a SoT of Tile$_{G2}$. In the case of unate functions, the layout technique is similar to CMOS style. As a generic example for unate functions, we map a 3-input OR function onto a pair of adjacent tiles (Fig. 21). Mapping of all the representative functions with embedded XOR/XNOR (F_3, F_7, F_{12}, and F_{13}) is shown in Figs. 9(b) and 22 (for F_{12}).

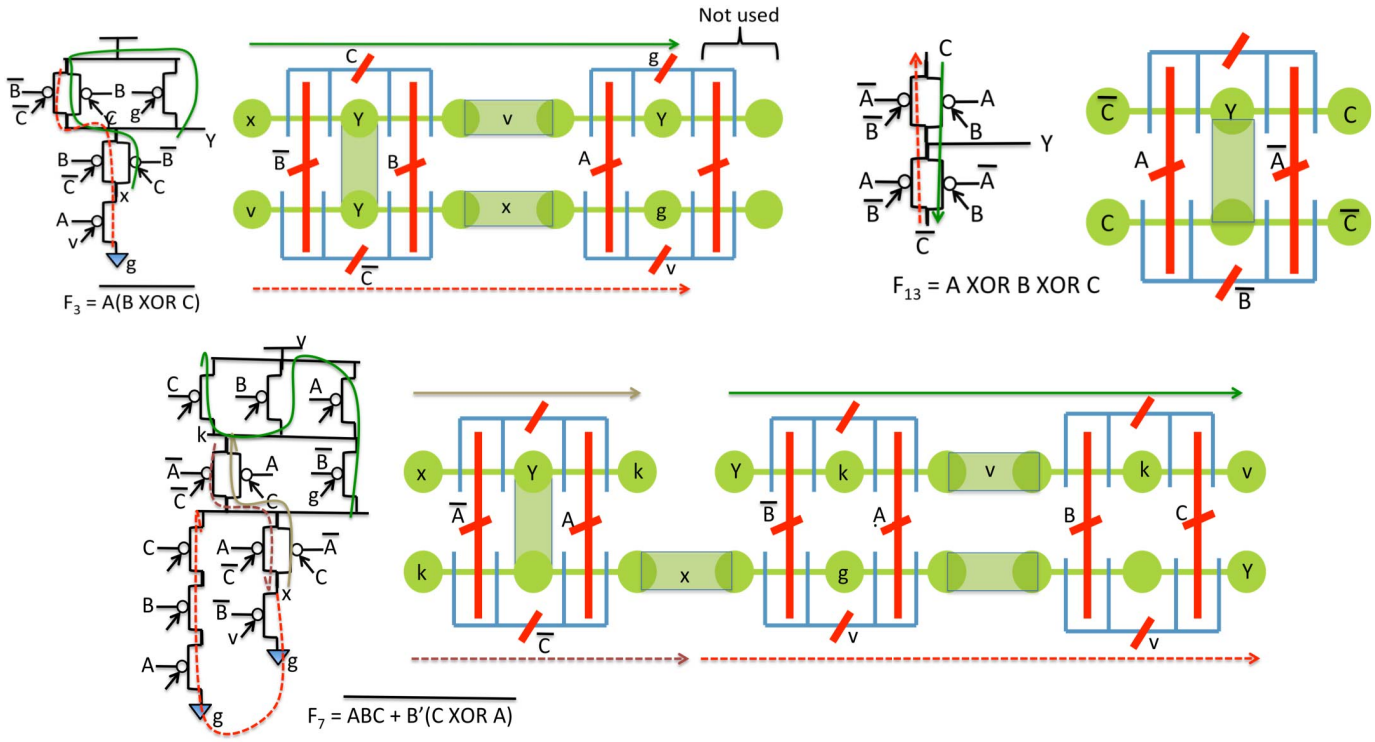

Fig. 22. Layout synthesis for all the 3-input binate functions (F_3, F_7, and F_{13}).

VI. Conclusion

The DG-SiNWFETs, with an extra PG, are promising contenders for efficient implementation of ambipolar logic. In this paper, we present an approach for designing an efficient regular layout fabric called SoT. To mitigate gate-level routing congestion caused by the extra PG, we propose a modeling technique based on DSDs and a topologic layout synthesis method for realizing Boolean functions with embedded XOR/XNOR functionality. By carrying technology mapping on SoT fabric, we show that tiles Tile$_{G1h2}$ and Tile$_{G2}$, on an average, outperform the one with Tile$_{G1}$ and Tile$_{G3}$ by 16% and 10% in area utilization, respectively. Finally, we present various case studies suggesting Tile$_{G1h2}$ and Tile$_{G2}$ as the basic building block for future ambipolar logic circuits.

Acknowledgment

The authors would like to thank Prof. Y. Leblebici, P.-E. Gaillardon, L. G. Amaru, and J. Zhang for their helpful; discussions. They would also like to thank M. De Marchi and D. Sacchetto for their insight on DG-SiNWFET technology.

References

[1] S. D. Suk *et al.*, "High performance 5 nm radius twin silicon nanowire MOSFET (TSNWFET): Fabrication on bulk Si wafer, characteristics, and reliability," in *Proc. IEEE IEDM*, Dec. 2005, pp. 717–720.

[2] R. M. Y. Ng, T. Wang, and M. Chan, "A new approach to fabricate vertically stacked single-crystalline silicon nanowires," in *Proc. IEEE Conf. EDSSC*, Dec. 2007, pp. 133–136.

[3] D. Sacchetto, M. H. Ben-Jamaa, G. De Micheli, and Y. Leblebici, "Fabrication and characterization of vertically stacked gate-all-around Si nanowire FET arrays," in *Proc. ESSDERC*, Sep. 2009, pp. 245–248.

[4] M. H. Ben Jamaa, D. Atienza, Y. Leblebici, and G. De Micheli, "Programmable logic circuits based on ambipolar CNFET," in *Proc. 45th ACM/IEEE DAC*, Jun. 2008, pp. 339–340.

[5] M. De Marchi *et al.*, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Dec. 2012, pp. 8.4.1–8.4.4.

[6] T. Uehara and W. M. Vancleemput, "Optimal layout of CMOS functional arrays," *IEEE Trans. Comput.*, vol. C-30, no. 5, pp. 305–312, May 1981.

[7] C.-Y. Hwang, Y.-C. Hsieh, Y.-L. Lin, and Y.-C. Hsu, "A fast transistor-chaining algorithm for CMOS cell layout," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 7, pp. 781–786, Jul. 1990.

[8] R. L. Maziasz and J. P. Hayes, "Layout optimization of static CMOS functional cells," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 7, pp. 708–719, Jul. 1990.

[9] J. Tejas *et al.*, "Maximization of layout printability/manufacturability by extreme layout regularity," *J. Micro/Nanolithogr.*, vol. 6, no. 3, pp. 0310111–0310115, Jul./Sep. 2007.

[10] Y.-W. Lin, M. Marek-Sadowska, and W. Maly, "Transistor-level layout of high-density regular circuits," in *Proc. ISPD*, 2009, pp. 83–90.

[11] Y. Ran and M. Marek-Sadowska, "Designing via-configurable logic blocks for regular fabric," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 1, pp. 1–14, Jan. 2006.

[12] B. Taylor and L. Pileggi, "Exact combinatorial optimization methods for physical design of regular logic bricks," in *Proc. 44th ACM/IEEE DAC*, Jun. 2007, pp. 344–349.

[13] F. Mailhot, "Technology mapping for VLSI circuits," Dept. Comput. Syst. Lab., Stanford Univ., Stanford, CA, USA, Tech. Rep. CSL-TR-94-646, Dec. 1994.

[14] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*. New York, NY, USA: Academic, 1985.

[15] S. Bobba, M. De Marchi, Y. Leblebici, and G. De Micheli, "Physical synthesis onto a Sea-of-Tiles with double-gate silicon nanowire transistors," in *Proc. 49th Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 42–47.

[16] A. Colli, S. Pisana, A. Fasoli, J. Robertson, and A. C. Ferrari, "Electronic transport in ambipolar silicon nanowires," *Phys. Status Solidi B*, vol. 244, no. 11, pp. 4161–4164, 2007.

[17] Y.-M. Lin, J. Appenzeller, J. Knoch, and P. Avouris, "High-performance carbon nanotube field-effect transistor with tunable polarities," *IEEE Trans. Nanotechnol.*, vol. 4, no. 5, pp. 481–489, Sep. 2005.

[18] A. K. Geim and K. S. Novoselov, "The rise of graphene," *Nature Mater.*, vol. 6, no. 3, pp. 183–191, 2007.

[19] I. O'Connor, J. Liu, D. Navarro, I. Hassoune, S. Burignat, and F. Gaffiot, "Ultra-fine grain reconfigurability using CNTFETs," in *Proc. 14th IEEE ICECS*, Dec. 2007, pp. 194–197.

[20] M. H. Ben Jamaa, K. Mohanram, and G. De Micheli, "Novel library of logic gates with ambipolar CNTFETs: Opportunities for multi-level logic synthesis," in *Proc. IEEE Conf. DATE*, Apr. 2009, pp. 622–627.
[21] A. Zukoski, X. Yang, and K. Mohanram, "Universal logic modules based on double-gate carbon nanotube transistors," in *Proc. 48th ACM/EDAC/IEEE DAC*, Jun. 2011, pp. 884–889.
[22] M. De Marchi, S. Bobba, M. H. Ben Jamaa, and G. De Micheli, "Synthesis of regular computational fabrics with ambipolar CNTFET technology," in *Proc. 17th IEEE ICECS*, Dec. 2010, pp. 70–73.
[23] C. Mead and L. Convey, *Introduction to VLSI Systems*. Reading, MA, USA: Addison-Wesley, 1979.
[24] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA, USA: Kluwer, 1984.
[25] R. H. Katz, *Contemporary Logic Design*. Redwood City, CA, USA: Benjamin Cummings, 1994.
[26] (2000). *Synopsys Design Compiler*. [Online]. Available: http://www.synopsys.com/Tools/Implementation/RTLSynthesis/DCGraphical/Pages/default.aspx
[27] *ABC: A System for Sequential Synthesis and Verification*. [Online]. Available: http://www.eecs.berkeley.edu/∼alanmi/abc/, accessed 2012.
[28] S. Muroga, *Threshold Logic and Its Applications*. New York, NY, USA: Wiley, 1971.

Shashikanth Bobba received the Ph.D. degree in developing new design methodologies and CAD tools for emerging nanotechnologies from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland.

He was a Visiting Researcher with Stanford University, Stanford, CA, USA, and CEA-LETI, Grenoble, France. He was with Agilent Technologies, Ghent, Belgium; Ericsson, Gothenburg, Sweden; and Telecom Italia, Turin, Italy. He is currently the Chief Technology Officer of 3D-EDA, Monolithic 3-D Inc., where he leads the Research Team developing EDA software for designing 3-D integrated circuits with monolithic 3-D technology. His research has led to over 25 international publications in leading conferences/journals, and holds three patents.

Dr. Bobba was a recipient of the Best Paper Award at the IEEE/ACM Nanoarch in 2012.

Giovanni De Micheli received the Nuclear Engineer degree from the Politecnico di Milano, Milan, Italy, in 1979 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 1980 and 1983, respectively.

He was a Professor of Electrical Engineering with Stanford University, Stanford, CA, USA. He is currently a Professor and the Director of the Institute of Electrical Engineering and the Integrated Systems Centre with the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He is also a Program Leader of the Nano-Tera.ch Program. He is interested in heterogeneous platform design, including electrical components and biosensors, and data processing of biomedical information. He authored a book entitled *Synthesis and Optimization of Digital Circuits* (New York, NY, USA: McGraw-Hill, 1994), and has coauthored and coedited eight other books, and over 600 technical articles. His h-index citation is 84 according to the Google Scholar. His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies, networks-on-a-chips, and 3-D integration.

Prof. De Micheli is a Fellow of the Association for Computing Machinery and a member of the Academia Europaea. He is also a member of the Scientific Advisory Board of imec, Leuven, Belgium; CfAED, Dresden, Germany; and STMicroelectronics, Geneva, Switzerland. He was the Chair of several conferences, including DATE in 2010, pHealth in 2006, VLSI SOC in 2006, DAC in 2000, and the International Conference on Computer Design in 1989. He has served IEEE in several capacities, namely, the Division 1 Director from 2008 to 2009, the Co-Founder and the President Elect of the IEEE Council on Electronic Design Automation from 2005 to 2007, the President of the IEEE CAS Society in 2003, and the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 1997 to 2001. He was a recipient of the IEEE Circuits and Systems Society (CAS) Mac Van Valkenburg Award for contributions to theory, practice, and experimentation in design methods and tools in 2012, the IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems in 2003, the Golden Jubilee Medal for outstanding contributions to the IEEE CAS Society in 2000, the D. Pederson Award for the Best Paper on the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS in 1987, and several best paper awards, including the Design Automation Conference (DAC) in 1983 and 1993, the Design Automation and Test in Europe (DATE) in 2005, and Nanoarch in 2010 and 2012.