Emerging Technologies: Challenges and Opportunities for Logic Synthesis

Alberto Bosio[1], Mayeul Cantan[1], Cedric Marchand[1], Ian O'Connor[1]
Petr Fiser[2], Arnaud Poittevin[1], and Marcello Traiola[1]
[1]*Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, 69130 Ecully, France*
[2]*Czech Technical University in Prague, Czech Republic*

***Abstract*—In computer engineering, logic synthesis is a process by which an abstract specification of desired circuit behavior is turned into a design implementation in terms of logic gates. Historically, logic synthesis was tightly related to the physical implementation of the logic gates. Nowadays, pushed by the forecasted end of Moore's law, several emerging technologies (e.g., nanodevices, optical computing, quantum computing) are candidates to either replace or co-exist with the *de facto* standard CMOS technology. The main consequence of the rising of those emerging technologies is that the logic synthesis has to face new issues and, at the same time, exploits new opportunities. The goal of this paper is thus to present three emerging technologies (Vertical Nanowire Field Effect Transistors, Ferroelectric Transistors, and Memristors), how to use them to implement logic gates, and the main challenges and issues for the logic synthesis.**

***Index Terms*—Emerging Technologies, Logic Synthesis, FeFETs, Vertical Nanowires, Memristors**

I. Introduction

Energy and computer efficiency is undoubtedly one of the major driving forces of current computer industry, which is relevant not only for supercomputers, but also for small portable personal electronics and sensors. However, today's computing architectures (mainly based on the CMOS technology) are facing major challenges making them unable to meet the requirements. Such challenges are: power wall, memory wall, and Instruction Level Parallelism wall. Moreover, even the dominating CMOS technology (which made manufacturing of computers feasible) is suffering, especially nodes below 20 nm. At this level the physical characteristics of such devices are leading to high static power consumption, reduced reliability; not to mention increased cost. All of these have led to saturated computer performance and the slowdown of the traditional device scaling, making today's computing systems unable to deliver the required computing and energy efficiency.

Due to these limitations, many alternative technologies being able to deliver the required demands at affordable cost are under investigation. CMOS based logic gates are well known and a holistic ecosystem of CAD tools is available to implement each step of the production flow: design, simulation, verification, synthesis, and test. On the other hand, emerging technologies may suffer from the inadaptation of existing CAD tools, especially for the synthesis. This paper aims at presenting three promising emerging technologies: Vertical Nanowire Field Effect Transistors, Ferroelectric Transistors, and Memristors. For each one, we show how to build logic gates and how those gates correspond to a challenge and/or opportunity for the synthesis.

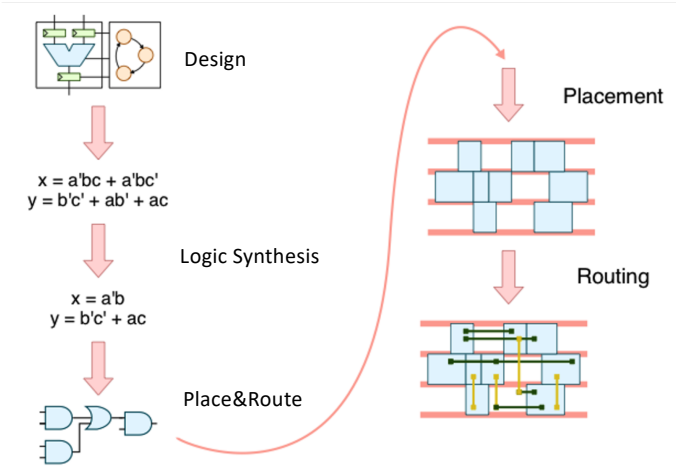

Fig. 1: Synthesis flow.

The paper is structured as follows. Section II provides the background of synthesis and in particular of logic synthesis. Section III presents the Vertical Nanowire Field Effect Transistors, Section IV presents the Ferroelectric Transistors and Section V the memristors. Finally, Section VI concludes the paper.

II. Logic Synthesis

In the digital circuits production flow, the Synthesis takes in input the circuit model description, usually expressed in a hardware description language (e.g., VHDL, Verilog) and produces as output the gate level netlist for a given layout floorplan. Fig. 1 details the synthesis and it is mainly focused on the two main steps: (i) Logic synthesis and optimization and (ii) Place&Route.

As defined in [1], the overall problem of logic synthesis is the one of finding "the best implementation" of a Boolean function. The term "best" corresponds to a trade-off between several metrics such as the area, delay, and power consumption. The Place&Route aims at optimizing the physical placement of each logic gate into a given layout floorplan and route the logic gate interconnections. It is important to keep in mind that logic synthesis is usually based on the knowledge of the technology used to implement logic gates. For

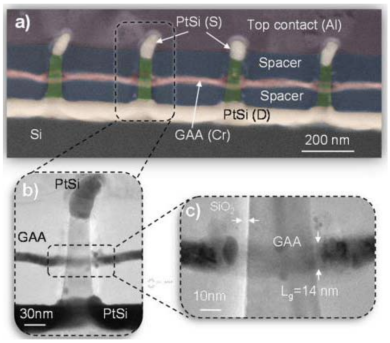

Fig. 2: VNWFET device [7]: (a) STEM image in cross section of the vertical transistor implemented in nanowire arrays, (b) single nanowire showing its (c) gate formation.

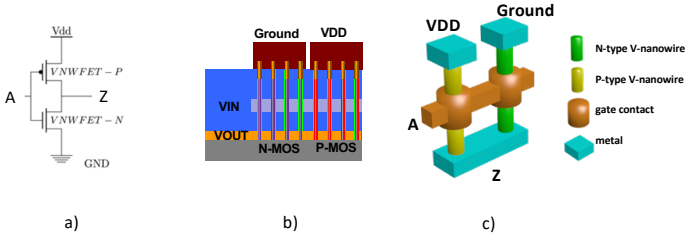

Fig. 3: (a) transistor netlist (b) layout cross-section (c) 3D layout representation.

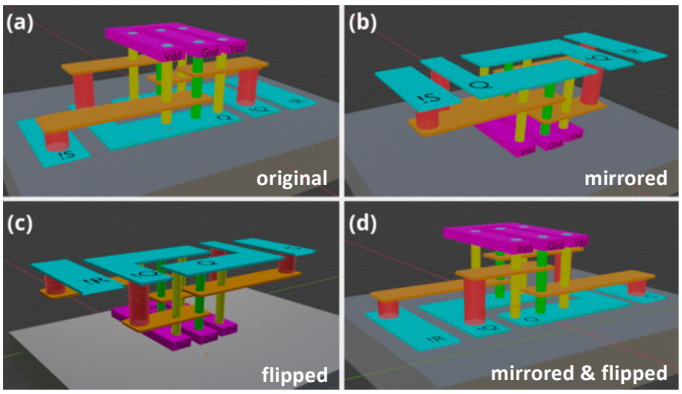

Fig. 4: Different 3D layout representations of the same logic gate.

example, first logic synthesis approaches addressed Boolean functions expressed in the Sum-of-Products (SoP) form and the optimization targeted to reduce the cardinality of logic covers (i.e., the number of product terms) [2]. This worked well when PLAs were used for the physical implementation of logic gates because they have rectangular shapes with rows associated with product terms. Hence, reducing the number of product terms reduces the area, thus leads to achieve the "best" implementation of the Boolean function. In the 1980', with the establishment of CMOS technology, logic gates and libraries of components paved the way for modern logic synthesis algorithms and tools [3], [4]. Here the problem consists of mapping Boolean functions into the "Best" interconnection of instances of library elements as depicted in Fig. 1.

Today, with the rising of alternative technologies to CMOS, we are facing new challenges for logic synthesis. It is therefore mandatory to well understand the logic gates built on the top of emerging technologies and identify the available opportunities.

III. Vertical Nanowire Field Effect Transistors (VNWFET)

This section explores preliminary logic gates based on VNWFET technology and the related challenges and opportunities for logic synthesis. Gate-all-around Vertical Nanowire Field Effect Transistors (VNWFET) are emerging devices, which are well suited to pursue scaling beyond lateral scaling limitations around 7nm. The VNWFET technology has a junctionless architecture composed of a homogeneous highly doped nanowire channel, patterned into boron doped (2 x 10^{19} cm^{-3}) Si substrate. The current flows between silicided source/drain contacts and is controlled by a gate-all-around structure with a physical channel length of 14nm as shown in Fig. 2. More details on the fabrication steps can be found in [5], [6].

Fig. 3 shows the implementation of the Inverter gate with two vertical transistors (P, and N type). In the figure, we present the classical transistor level view (Fig. 3a) and the layout cross-section (Fig. 3b) in which we can note the presence of two nanowire arrays (one array for the P- and the other for the N-channel). The 3D representation is shown in Fig. 3c.

Vertical transistor channels lead to a paradigm change in the design of logic cells, indeed, they further improve the gain in circuit density. Moreover, the third dimension enables numerous spatial configurations for the same logic functionality [8]. An example is given in Fig. 4 where the four pictures show different layouts of the same SR latch. This new degree of freedom is clearly an opportunity for the back-end (i.e., Place&Route) synthesis.

More complex gates can be easily implemented. We actually investigated the possibility to implement compact gates and, at the same time, able to perform complex Boolean functions. The first result we got is shown in Fig. 5. Here we designed a compact gate composed of 2P and 2N vertical transistors. The gate has 5 inputs (from A to E). We simulated it using the model of [6], and we got the following Boolean function:

$$Z = E(\overline{A} \cdot \overline{C} + BC + \overline{B} \cdot \overline{D} + AD) \quad (1)$$

Now how can these gates be fully exploited by logic synthesis? One opportunity can be explored with 3-input gate logic synthesis [9]. In that paper, authors investigated advantages of 3-input gates as constituents of logic networks. They found out that some 3-input gates can be extremely powerful in efficient representing a Boolean function (i.e. the *dot* gate in [9]). Coming back to our gate of Fig. 5, we can split the E input, actually common to the four vertical transistors into four individual inputs E, F, G, and H (one for each

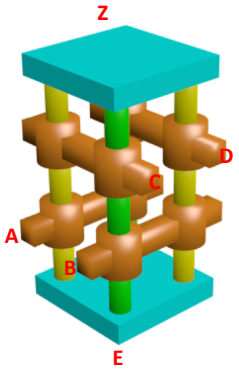

Fig. 5: 2P2N Vertical transistors based gate.

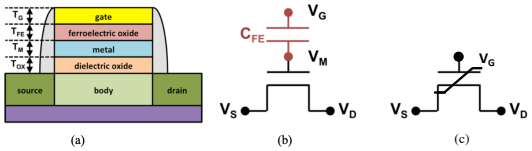

Fig. 6: (a) Ferroelectric HfO_2 device gate stack (b) equivalent circuit schematic (c) electronic symbol.

vertical transistor). It is now possible to consider the following subgroups of 3-inputs depending on the vertical transistor type:

- $Z = f(EAC)$ for P-type;
- $Z = f(FBC)$ for N-type;
- $Z = f(GAD)$ for P-type;
- $Z = f(HBD)$ for N-type.

In other words, it is possible to explore new 3-input gates to leverage logic synthesis. Finally, also the Place&Route synthesis can exploit the high number of possible 3D layouts to provide a more dense circuit (reducing area and volume overhead).

IV. Ferroelectric Transistor Based Logic

From a structural point of view, a ferroelectric transistor (FeFET) is simply an extension of a regular bulk or FDSOI (Fully Depleted Silicon On Insulator) MOSFET with an additional layer of ferroelectric HfO_2-based material [10] inside the gate stack as depicted in Fig. 6.

The ferroelectric layer behaves as a ferroelectric capacitance C_{FE} between V_G and V_M which actually controls the state of the FET channel. FeFETs operate in two different modes: a non-volatile mode, which requires hysteretic operation, and a steep switching mode, which can be hysteretic or non-hysteretic [10]. The ratio between the ferroelectric capacitance and the dielectric capacitance determines the FeFET operation mode. In this paper we target the **non-volatile mode** and we show how FeFET can be exploited to design non-volatile logic gates: Nand and *Configurable* gate.

All the presented circuits have been designed using 28nm technology provided from GLOBALFOUNDRIES and simulated using Cadence Virtuoso. Simulations are based on

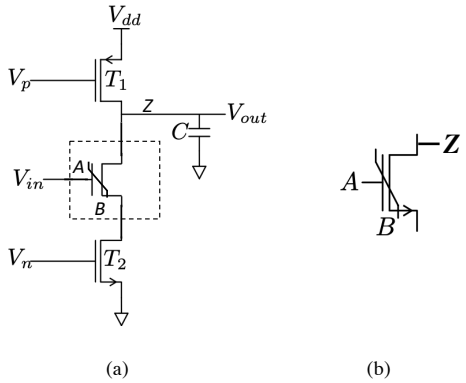

Fig. 7: (a) Full Schematic view of the FeFET based Nand, (b) simplified view

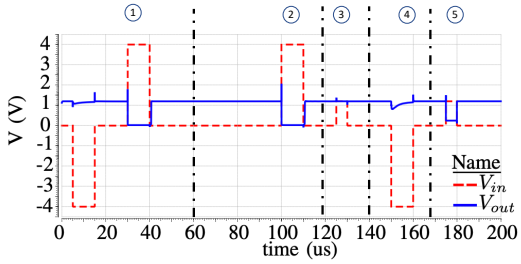

Fig. 8: Simulation of the FeFET Dynamic Nand gate

Preisach-based FeFET model [11] which has been fitted on ASIC characterization [12].

A. Non-Volatile Nand Gate

This Nand gate is the most simple one since it is implemented by using a single FeFET transistor as shown in Fig. 7a. The schematic corresponds to a dynamic logic gate [13] in which the FeFET (dashed box) implements the Nand. Fig. 7b provides a simplified view of the Nand gate in terms of inputs A, B and output $Z = \overline{A \cdot B}$.

One of the two operands, in our case B, is a constant value stored as a charge in the ferroelectric capacitance. The second operand A, corresponds to the input voltage on the gate FeFET. The protocol to access the Nand gate is composed of the following 4 steps:

1) Write the constant operand B in the FeFET (details about FeFET writing protocol are provided in [10]);
2) Charge V_{out} to V_{dd}: V_p = V_{dd} (i.e., logic '1'), V_n = gnd (i.e., logic '0');
3) Activate the FeFET: V_p = gnd (i.e., logic '0'), V_n = V_{dd} (i.e., logic '1');
4) Set V_{in} to the voltage level corresponding to operand A;
5) Depending on A and B the FeFET will be closed or open. If open, the output capacitance C will be discharged through the FeFET leading to have V_{out} = gnd (i.e., Z = '0'). If close, the output capacitance C will maintain V_{out} = V_{dd} (i.e., Z = '1').

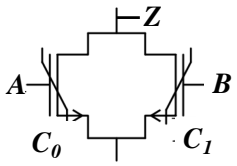

Fig. 9: Configurable gate

Fig. 8 shows a simulation waveform of the Nand gate. For the sake of simplicity, we only report the signals V_{in} and V_{out}. The simulation starts with the FeFET initialisation (①) by applying two consecutive opposite write pulses. In our simulation, a negative pulse writes a logical '1' inside the FeFET, while a positive pulse writes a logical '0'. Note that the initialisation has to be done only one time. In ② we set B = '0' (positive pulse applied to V_{in}) and in ③ we set A = '1' ($V_{in} = V_{high}$). In ③, the output Z = '1' ($V_{out} = V_{high}$). The second simulated pattern starts in ④ by setting B = '1' (negative pulse applied to V_{in}). In ⑤, we set A = '1' ($V_{in} = V_{high}$) and the output Z = '0' ($V_{out} = V_{low}$).

B. Configurable Gate

Fig. 9 depicts the schematic of a gate that can be configured to implement different Boolean functions depending on the logic value stored in the FeFETs. First of all, the output Z is set to logic '1' if and only if the two transistors are opened (i.e., not passing). This can be formalize as following:

$$Z = \overline{A \cdot C_0} \cdot \overline{B \cdot C_1} \tag{2}$$

where C_0 and C_1 are the values stored in the FeFET and A, B correspond to the gate inputs. Based on the Eq. 2, it is possible to configure the gate to implement several Boolean functions as shown in Table I. Moreover, if we set $C_0 = \overline{B}$ and $C_1 = \overline{A}$ we can rewrite Eq. 2 as:

$$Z = \overline{A \cdot \overline{B}} \cdot \overline{B \cdot \overline{A}} \tag{3}$$

From Eq. 3 we can apply De Morgan's theorem and write:

$$\begin{aligned} Z &= \overline{\overline{A \cdot \overline{B}} \cdot \overline{B \cdot \overline{A}}} \\ &= \overline{(A \cdot \overline{B}) + (B \cdot \overline{A})} \\ &= \overline{A \oplus B} \end{aligned} \tag{4}$$

We can thus obtain the **Xnor** gate as shown in Eq.4. Using the same approach, we can obtain the **Nor** gate when $C_0 = C_1$ ='1' (last row of Table I)

$$\begin{aligned} Z &= \overline{A} \cdot \overline{B} \\ &= \overline{A + B} \end{aligned} \tag{5}$$

To summarize:

1) **Nand Gate** can be exploited to synthesize any Boolean function in which some inputs are constant values stored in the FeFET gates. For example in the case of filter coefficients.
2) **Configurable Gate** can be exploited to synthesize any Boolean function. Moreover, the same circuit can be configured to execute different Boolean functions as for the polymorphic circuits [14].

TABLE I: Possible configuration

C_0	C_1	Z
0	0	1
0	1	$\overline{B}$
1	0	$\overline{A}$
1	1	$\overline{A} \cdot \overline{B}$

(a) SET (b) RESET

Fig. 10: Set and Reset operations [17]

V. Memristors based logic

A memristor is a non-linear bipolar device, characterized by an electrical resistance that is not constant but rather depends on the past history of current that has flowed through the device itself [15]. We resort to the memristor proposed in [16], where depending on the applied voltage value V_{in} applied to the memristor terminals, it is possible to *set* the resistance to a given value. We resort to the Snider Boolean Logic (SBL) convention [16] where a lower resistance R_{ON} represents the logic '0', while the higher resistance R_{OFF} represents the logic '1'. More in particular, we can define the following operations:

- **SET**: $V_{in} > V_{th}$, the memristor is set to logic '0' V_{in} is defined as V_w in Fig. 10a
- **RESET**: $V_{in} < -V_{th}$, the memristor is set to logic '1' V_{in} is defined as $-V_w$ in Fig. 10b
- **Read**: $gnd < V_{in} < V_{th}$, the memristor does not change its resistance. The current flowing through its terminal is proportional to its actual resistance vale (i.e., R_{ON} or R_{OFF}). V_{in} is defined as V_r

A. Fast Boolean Logic Circuits

The logic circuit implementation that we consider in this paper is Fast Boolean Logic Circuit (***FBLC***) [17]. The FBLC logic circuit implementation requires that the Boolean function is expressed in the SoP format:

$$M_1 + M_2 \ldots + M_n = \overline{\overline{M_1} \cdot \overline{M_2} \ldots \overline{M_n}} \tag{6}$$

where M_i are minterms. The left member of the Equation 6 can be easily manipulated through transformation rules (i.e., De Morgan's laws). The obtained form (right member of

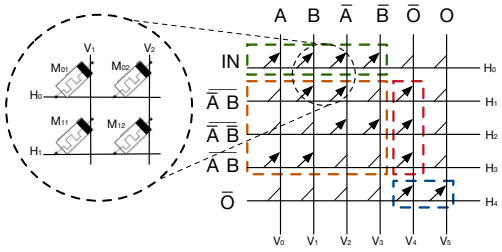

Fig. 11: FBLC Example

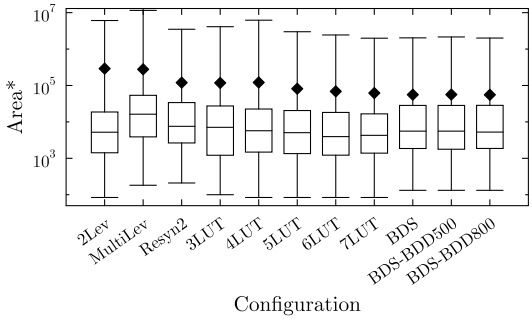

Fig. 12: Area values distribution

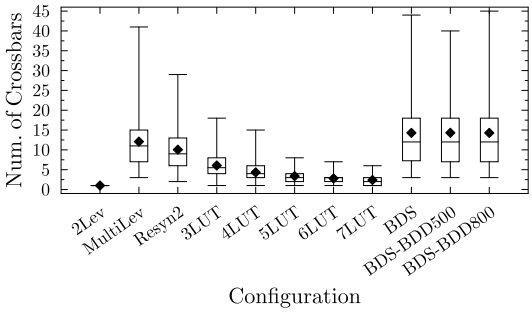

Fig. 13: Delay values distribution

the Equation 6) can be computed exploiting three Boolean operations: Nand, And and Not. Let us consider the following example. The Boolean function is defined as:

$$O = AB + \overline{A}B + \overline{AB} = \overline{\overline{AB} \cdot \overline{\overline{A}B} \cdot \overline{A\overline{B}}} \qquad (7)$$

Eq. 7 is implemented in the memristors crossbar array shown in Fig. 11. In order to properly drive signals to the data path, an external control unit has to perform the following phases.

1) Al the memristors in the crossbar are initialized to R_{OFF} (logic value '1');
2) Input values are copied to the input block (IN) memristors (first line of the example crossbar);
3) Values from IN are copied to to the Minterms blocks (Rows 2, 3 and 4 of the example crossbar);
4) Minterms are evaluated performing all the Nand operations in parallel;
5) Results are stored in the And block (column O)
6) The AND operation is performed and the result is stored in the Output block (last row of the example crossbar).

B. Experiments

For a given a Boolean function to be synthesized, we exploited different logic minimization approaches available in the state-of-the-art of synthesis tools, namely ABC [18], SIS [19] and BDS [20]. In particular, SIS has been exploited for the 2-level minimization, ABC for the multi-level minimization based on two-input ANDs and Inverters and on Look Up Tables (LUTs) mapping, while BDS for the multi-level minimization based on MUX and XOR.

Specifically, as for the 2-level minimization, we leveraged the *collapse* option embedded in SIS. Then, we exploited different commands embedded in ABC and BDS for obtaining different forms of multi-level functions. As for BDS, we used it with (i) limitSize 500 and (ii) limitSize 800. The LimitSize option sets the largest BDD size on which exact variable reordering can apply (number of variables times number of nodes in a BDD); the default is 200. For ABC, we resorted to the simple *strash* command (we refer to it as *MultiLev*) and to the *resyn2* script. Moreover, we also investigated the impact of a technology mapper. ABC embeds the FPGA mapping for obtaining a set of K-LUTs where K is the number of inputs of each LUT. We exploited the *if* command with the *-K* option to perform 3-7 FPGA LUTs mapping (thus K varies from 3 to 7). The main difference between those minimization options is the resulting number of required crossbars to be used to implement the Boolean function.

Then, we leveraged XbarGen (an in-house tool [21], [22]) for obtaining statistics (i.e., Area and Delay). We carried out experiments over about 300 different combinational circuits given from [23]–[26] with up to 1763 inputs and 2048 outputs and up to 40 thousands general terms. It is worth to mention that some circuits were obtained by extracting the combinational logic of sequential circuits.

Fig. 12 compares the area distribution of circuits synthesized exploiting the previously described logic optimizations (values are expressed on the logarithmic scale). As shown, synthesizing on crossbars composed by 6/7 inputs (6/7 LUT) leads to a reduction of the area overhead compared with previous configurations because the crossbars are denser. Specifically, the occupied area of 6/7 LUT configurations is comparable with the 2Lev configuration area. On the other hand, Fig. 13 shows that, on average, all configurations always lead to circuits with higher delay (i.e., they have more crossbars), w.r.t. the 2Lev configuration (i.e., a single crossbar). However, one big drawback of the 2Lev minimization is its limited scalability. For functions with many inputs (> 20), the resulting two-level form is prohibitively large. A good compromise is the LUT-based mapping. It significantly reduces the delay compared with the simple multi-level approach. Indeed, the delay of LUT-based implementations is very close to the 2-level mapping. We can thus consider the LUT-mapping as the best implementation considering area/delay characteristics.

Finally, as for logic syntheses obtained by means of BDS tool [20], the results were not impressive, as shown in the Figures. Indeed, it produces circuits with lots of crossbars (i.e., high delay) and area overhead levels comparable with the 5LUT configuration.

To summarize, we presented how of well-known logic optimization techniques such as the 2-level and multi-level optimizations can fit with FBLC memristor based logic synthesis. Contrary to the common implementation technology which makes use of simple logic gates, for memristor-based crossbars the area/delay curve is surely not trivial. Classic CMOS-oriented tools produce bad optimizations of logic functions for memristor-based crossbars, while by introducing a proper optimization (i.e., K-LUTs), we managed to restore the well-known area/delay trend.

VI. Conclusions

In this paper, we presented three emerging technologies (Vertical Nanowire Field Effect Transistors, Ferroelectric Transistors, and Memristors). For each of these technologies, we presented how to design logic gates and what are the challenges and opportunities for logic synthesis. We have shown that vertical transistors can cope well with the 3-input logic synthesis and they offer new opportunities for the Place&Route. Ferroelectric Transistors allow to implement non-volatile logic gates. Here one operand is stored inside the gate offering, from one side, opportunities for reducing the need of moving data, but also challenges since it is possible to design polymorphic gates. Finally, for the memristors based FBLC logic we tested several state-of-the-art logic synthesis approaches on a large set of circuit benchmarks. The results have shown that some of the existing approach may be useful, but globally, the logic synthesis community has a lot of opportunities to explore, thanks to the presented emerging technology based logic.

Acknowledgment

The authors acknowledge the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics", the LEGO project through ANR funding (Grant ANR-18-CE24-0005-01) and the European Union's Horizon 2020 research and innovation program under grant agreement No 780302 "3$\epsilon Ferro$".

References

[1] E. Testa, M. Soeken, L. G. Amar, and G. D. Micheli, "Logic synthesis for established and emerging computing," *Proceedings of the IEEE*, vol. 107, pp. 165–184, Jan. 2019.

[2] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Springer US, 1984.

[3] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1996.

[4] S. Hassoun and T. Sasao, *Logic Synthesis and Verification*. Kluwer Academic Publishers, 2002.

[5] Y. Guerfi and G. Larrieu, "Vertical silicon nanowire field effect transistors with nanoscale gate-all-around," *Nanoscale Research Letters*, vol. 11, Apr. 2016.

[6] C. Mukherjee, M. Deng, F. Marc, C. Maneux, A. Poittevin, I. O'Connor, S. L. Beux, C. Marchand, A. Kumar, A. Lecestre, and G. Larrieu, "3D logic cells design and results based on vertical NWFET technology including tied compact model," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*, pp. 76–81, 2020.

[7] G. Larrieu and X.-L. Han, "Vertical nanowire array-based field effect transistors for ultimate scaling," *Nanoscale*, vol. 5, no. 6, p. 2437, 2013.

[8] V. Moroz, X. Lin, L. Smith, J. Huang, M. Choi, T. Ma, J. Liu, Y. Zhang, J. Kawa, and Y. Saad, "Power-performance-area engineering of 5nm nanowire library cells," in *2015 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pp. 433–436, 2015.

[9] D. S. Marakkalage, E. Testa, H. Riener, A. Mishchenko, M. Soeken, and G. De Micheli, "Three-input gates for logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, p. 5, 2020.

[10] I. O'Connor, M. Cantan, C. Marchand, B. Vilquin, S. Slesazeck, E. T. Breyer, H. Mulaosmanovic, T. Mikolajick, B. Giraud, J. Noël, A. Ionescu, and I. Stolichnov, "Prospects for energy-efficient edge computing with integrated HfO2-based ferroelectric devices," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 180–183, 2018.

[11] K. Ni, M. Jerry, J. A. Smith, and S. Datta, "A circuit compatible accurate compact model for ferroelectric-FETs," in *2018 IEEE Symposium on VLSI Technology*, pp. 131–132, 2018.

[12] E. T. Breyer, H. Mulaosmanovic, S. Slesazeck, T. Mikolajick, and T. Mikolajick, "Demonstration of versatile nonvolatile logic gates in 28nm HKMG FeFET technology," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018.

[13] J. P. Uyemura, *Dynamic Logic Circuit Concepts*, pp. 287–347. Boston, MA: Springer US, 2001.

[14] A. Crha, R. Ruzicka, and V. Simek, "Synthesis methodology of polymorphic circuits using polymorphic NAND/NOR gates," in *2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*, pp. 612–617, 2015.

[15] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, pp. 507–519, September 1971.

[16] G. Snider, "Computing with hysteretic resistor crossbars," *Applied Physics A*, vol. 80, 2005.

[17] L. Xie, H. A. D. Nguyen, M. Taouil, S. Hamdioui, and K. Bertels, "Fast boolean logic mapped on memristor crossbar," in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 335–342, Oct 2015.

[18] A. Mishchenko *et al.*, *ABC: A system for sequential synthesis and verification*. 2012.

[19] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: a system for sequential circuit synthesis," No. UCB/ERL M92/41, 1992.

[20] C. Yang and M. Ciesielski, "BDS: a BDD-based logic optimization system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 866–876, Jul 2002.

[21] M. Traiola, M. Barbareschi, A. Mazzeo, and A. Bosio, "XbarGen: a memristor based boolean logic synthesis tool," in *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, Sep. 2016.

[22] M. Traiola, M. Barbareschi, and A. Bosio, "Estimating dynamic power consumption for memristor-based cim architecture," *Microelectronics Reliability*, vol. 80, pp. 241 – 248, 2018.

[23] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934 vol.3, May 1989.

[24] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *Design Test of Computers, IEEE*, vol. 17, pp. 44–53, Jul 2000.

[25] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," 1991.

[26] S. Yang, "Logic synthesis and optimization benchmarks," Dec. 1988.