HAL
open science

Logic Synthesis for Silicon and Beyond-Silicon Multi-gate Pass-Logic Circuits

Valerio Tenace, Andrea Calimera, Enrico Macii, Massimo Poncino

To cite this version:

Valerio Tenace, Andrea Calimera, Enrico Macii, Massimo Poncino. Logic Synthesis for Silicon and Beyond-Silicon Multi-gate Pass-Logic Circuits. 24th IFIP/IEEE International Conference on Very Large Scale Integration - System on a Chip (VLSISOC), Sep 2016, Tallinn, Estonia. pp.60-82, 10.1007/978-3-319-67104-8_4 . hal-01675195

HAL Id: hal-01675195

https://inria.hal.science/hal-01675195v1

Submitted on 4 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Logic Synthesis for Silicon and Beyond-Silicon Multi-Gate Pass-Logic Circuits

Valerio Tenace, Andrea Calimera, Enrico Macii, and Massimo Poncino

Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca degli Abruzzi 24, Torino, Italy
{valerio.tenace,andrea.calimera,enrico.macii,massimo.poncino}@polito.it

Abstract. In the last decade several new technologies have been proposed as possible replacement for MOSFETs; Silicon Nanowires, Magnetic Tunnel Junctions, Graphene p-n Junctions are just some of the most representative examples. Although their intrinsic differences, they all share a common key characteristic, i.e., enable the implementation of logic gates with an expressive power much higher than that of state-of-art silicon CMOS gates. This may translate into more complex and faster switching functions that count less devices. The view of new materials that can serve as technological vehicles for energy efficient circuits and systems attracted the interested of the whole electronics research community. Apart from the many technological aspects, the path towards large-scale integration of emerging devices crosses the need of *(i)* new integration strategies that better fit the characteristics of the new technologies and *(ii)* new computer-aided design (CAD) methodologies able to cope with the complexity of today's design specs. The availability of this two elements may open the way for fast design space exploration and better assessment of new technologies against standard CMOS.
This work focuses on logic synthesis and optimization tools for ultra-low power pass-gate circuits mapped into emerging technologies, Graphene and silicon nano-wires. More specifically, we describe a novel multi-function decomposition engine that *(i)* efficiently performs abstract circuit modeling through a highly-compact data structure called Multi-Function Pass Diagram (MFPD), *(ii)* provides an effective multi-gate synthesis&optimization flow, *(iii)* allows accurate power/delay estimations. The contents reported in the following sections represent one of the first examples of how dedicated algorithms and data-structures can substantially improve the quality-of-design when moving from CMOS to emerging technologies.
Simulation run conducted on different benchmarks demonstrate that pass-gate circuits synthesized with the proposed tool are smaller and shallower, hence less power hungry and faster than circuits obtained through conventional synthesis methodologies based on standard design flows. As an additional contribution, the results prove that our solution is not only applicable to beyond-silicon technologies but also to standard MOSFETs.

Keywords: Emerging technologies, Graphene, Silicon nano-wires, Pass-Gate Logic, CAD, Logic Synthesis, Low-Power, Adiabatic Computing

1 Introduction

1.1 CMOS at the end of the line

The introduction of Metal-Oxide-Semiconductor Field-Effect-Transistors (MOSFETs), officially set in 1947 at Bells Labs as a replacement to vacuum tubes, represents a milestone in the industry of semiconductors. Since then, and after 60-year long research efforts, Complementary MOS (CMOS) electronic circuits have become the dominant technology for the entire ICT segment.

The continuous demand for smaller, faster and more power-efficient Integrated Circuits (ICs) have pushed CMOS technology close to its boundary. At the time being, technology trends are clearly highlighting that a radical shift in thinking digital hardware design might come soon. Among the many aspects, we highlight three well known issues that sustain this claim.

- **Non-ideality of the Silicon scaling process**: below the 45nm node the technology scaling process faced several limitations due to (*i*) the increased difficulty in discretizing transistors on a physical die, a problem related to the gate-oxide thickness that is slowly approaching a few-atom width [1]; (*ii*) the miniaturized gate length of MOS transistors and the upsurge of short-channel effects (SCEs), leakage current in particular, which represent a serious reliability issue [2]; (*iii*) as transistors size decreases, power dissipation and process-variation induced reliability issues become critical [3]. These issues impact the fabrication yield of reliable ICs.
- **Nanometric CMOS styles are no longer the most energy-efficient integration strategy**: static CMOS has been taken as a reference style for mainstream VLSI circuits due to high noise immunity, resilience to supply-voltage scaling and low leakage currents. However, as the technology scaling process went below the 90nm mark, some of these characteristics faded out due to SCEs. This suggests that other logic families that were discarded in the past, e.g., Dynamic-Logic, Pass-Transistor-Logic [4], may represent a new way out for low-power ICs.
- **Hitting the power-wall and the dark-silicon problem**: achieving ultra-low power consumptions is becoming a vital feature for consumer electronics, especially in the context of the Internet-of-Things (IoT) [5] where always-on, always-connected devices running sensoring applications and data-intensive computing represent the new mainstream paradigm. Even if many low-power techniques for CMOS circuits and systems are available today [6], e.g., Dynamic-Voltage-Frequency Scaling, Power-Gating, Multi-Threshold-Voltage and Reverse-Body-Biasing, the power consumed by CMOS based Systems-on-Chips (SoCs) architectures may exceed the power budget. This implies that an ever larger portion of the silicon die must be kept off (Dark-Silicon). To address this issue, standard SoC architectures will make space to less power hungry solutions that implies the use of dedicated accelerators with embedded memory resources and more energy efficient circuitry.

For such reasons, soon or late, Silicon, CMOS and standard Von-Neumann architectures will drop the scepter in favor of emerging technologies, alternative integration strategies and new architectures. From a technological point of view, recent works proposed several options such as Ambipolar Silicon-NanoWires [7], Graphene p-n Junctions [8], Graphene Nanoribbons[9], Magnetic Tunnel Junctions [10] and Domain-Wall Nanowires [11]. Apart from their improved electrical characteristics, those technologies could enhance switching primitives with new fascinating properties able to accommodate the specifications of alternative computing paradigms.

1.2 Candidates to replace the CMOS technology

The above qualitative analysis suggests that the Silicon/CMOS pair could be soon replaced by some new material and a more energy efficient integration strategy. Among the many options we believe ambipolar technologies, such as Graphene or Silicon-Nanowires, integrated à la *pass-gate logic* style, a.k.a. *pass-transistor logic* (PTL), represent an interesting option. In particular, as it will be shown later in the text, Graphene-based devices are particularly suited for pass-gate logic.

The choice of PTL is justified by its high intrinsic efficiency, already proven for silicon technologies. PTL circuits can implement logic functions with a lower transistor count, smaller parasitic capacitance and hence better performance [12]. Even today's CMOS libraries make use of PTL for some logic gates, e.g., flip-flops and multiplexers, because of their efficient implementation. Moreover, PTL circuits offer an opportunity to work "adiabatically", namely, mimicking the adiabatic (i.e., without energy exchange) charging process [13]. Adiabatic PTL may find space with the implementation of dedicated hardware accelerators in charge of processing "slow" physical-data (e.g., biometric signals) with a very limited energy budget [5].

The use of PTL and, more precisely, adiabatic PTL, has been already proven for emerging technologies, such as nanoelectromechanical switches (NEMs), carbon nanotubes (CNTs), and graphene p-n junctions. For such devices the PTL style enables the design of logic circuits with improved energy efficiency if compared to CMOS [14,15,16].

1.3 Lack of logic synthesis tools for PTL

It is clear that new hardware schemes, such as PTL, will inevitably ask for new CAD tools for the logic synthesis of digital blocks. Algorithms and data structures for the logic synthesis evolved following the growth of semi-custom CMOS libraries, while synthesis for PTL has been improved only marginally. This is why, even today, PTL remains underutilized [12]. It's not a coincidence that most of the previous works do focus on circuits for very specific arithmetic functions [17,18] or handcrafted basic Boolean logic gates [4,19]. Indeed, when the target design turns into random logic, standard multi-level synthesis engines

can't exploit the structural properties of PTL. That brings to sub-optimal implementations that typically require *ad-hoc* actions at the post-synthesis stage.

This problem is not new to the research community and several solutions have been introduced in the last years. Most of them, if not all, are closely related to the concept of Binary Decision Diagrams (BDDs) or some of its variants [20,12,21]. There are two main reasons behind the use of BDDs. First, there exists a one-to-one matching between the BDD representation of the logic function and the final PTL circuit implementation; this enables the concept of *one-pass logic synthesis* [22] where logic optimization and technology mapping are carried out concurrently on the same data structure thereby saving CPU and memory usage. Second, BDDs [23] are a very mature data-structure with lots of available optimization algorithms for redundancy removal and circuit optimization.

Despite the efficiency of BDDs as data-structure is unquestionable, BDD-based synthesis tools show many limitations. First, the tree-like structure of BDDs reflects into a deep circuit topology with large depth, and hence large propagation delays. Second, state-of-the-art decomposition methods for BDDs construction all operate using a pre-fixed variable-order (VO), namely, the order used for variable expansion is fixed during the entire decomposition procedure, no matter what the logic function is. Since VO affects the vertex-set cardinality of BDDs, a wrong VO might result into dramatic area increase of the resulting circuit. Third, decomposition methods are constrained to a "single-function" decomposition. Such a function, here referred as $g(\mathbf{X})$, differs depending on the type of BDD variant in use, e.g., MUX for standard BDDs [23], XOR for Biconditional-BDDs [21]. Logic circuits dominated by $g(\mathbf{X})$, e.g., XOR-rich arithmetic circuits, take advantage of this characteristic, others, like random logic circuits, may suffer from sub-optimal minimization. While the first two issues have been addressed in [24] with the introduction of the *Pass Diagram* (PD) data-structure and the non-fixed VO decomposition, this work elaborates on the third issue, i.e., how to overcome "single-function" decomposition.

1.4 Contribution of this work

As an extension of the contents proposed in [25], this work gives a comprehensive description of efficient abstract models and data-structures that are particularly suited for the synthesis of Multi-Gate Pass Logic (MGPL) circuits mapped with emerging technologies, Graphene and Silicon-NanoWires in particular, or, alternatively, with standard silicon MOSFETs.

An MGPL circuit consists of series connections of two-input *pass-gates* that can be turned-ON (OFF) and thus open (close) an electrical path between a clocked-power (the source) and the main output (the leaf); multiple paths are connected in parallel making the final logic circuit. Hence, similar to PTL, the information is not carried in the form of charges stored in parasitic capacitance, but rather through the root-to-leaf propagation of the clock-power signal. It is worth noticing that, differently from any other existing PTL solutions, MGPL makes use of pass-gates that embed multiple Boolean operators, like AND, OR,

NAND, and not just MUX or XOR as in the previous works; the choice of which operators depends on the technology in use.

We introduce a novel abstraction model, namely, the binary *Multi-Function Pass-Diagram* (MFPD), a graph-based representation for k-ary Boolean functions. An MFPD is a polarized, acyclic directed graph made up of N root-to-leaf logical paths. Each path is composed of an arbitrary number of two-input nodes connected in series, where each node represents a binary connective between two, out of k, primary input variables and can assume either a TRUE logical value, e.g., closed switch, or a FALSE logical value, namely an open switch. Under a specific input pattern, logical paths can be activated (all nodes are closed switches) in mutual exclusion (1 path out of N) and thus create a *gateway* from the root to the leaf. In such case, the equivalent logic function represented by the MFPD is evaluated as TRUE; on the contrary, when no active paths do exist, the logic function is said to be evaluated as FALSE. This structure matches the topology of a Multi-Gate Pass-Logic circuit.

The construction of a binary MFPD encompasses two major steps: *multi-function decomposition* using a set of basic Boolean operators, e.g., AND, OR, XOR and their complement; and *redundancy removal* through iterative reduction rules. Those phases have been integrated into an automatic synthesis and optimization tool named *Kanon*. Moving from single- to multi-function decomposition can be conceptually seen as the shift from two-level to multi-level synthesis carried out for CMOS circuits.

We apply our tool *Kanon* to a sub-set of generic benchmarks mapped onto three different technologies, i.e., Silicon MOS transistors, Ambipolar Silicon Nanowires and Graphene p-n junctions. The use of generic benchmarks avoids biased results due to the presence of circuits dominated by a specific function, the use of different technologies demonstrates that the proposed solution well fits both silicon and beyond-silicon technologies. The obtained MGPL circuits are then compared against standard PTL circuits synthesized using state-of-art BDD-based tools. The collected results validate the functionality of the proposed MFPD model and the related multi-function decomposition, whilst simulations using SPICE models quantify the energy efficiency of MGPL circuits.

2 Multi-Gate Pass Logic

A first example of pass-logic circuit for emerging devices has been recently proposed in [26] in the form of Pass-XNOR Logic (PXL) network using graphene p-n junctions. A PXL circuit consists of a network of Pass-XNOR Gates (PXGs); PXGs connected in series form a logic path, while logic paths connected in parallel connect the root of the circuit (fed by a clock-power signal) to the leaf (the main output). The clock-power signal works as an evaluation signal that eventually reaches the output when at least one parallel logic path is ON; in this case the logic function is evaluated as TRUE, i.e., 1-logic. When none of the parallel logic paths is ON, the propagation of the clocked-power signal is inhibited and the logic function is evaluated as 0-logic.

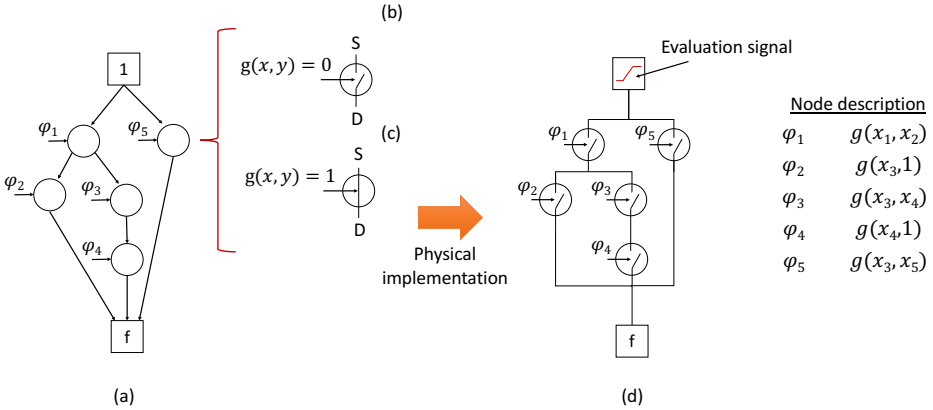

Fig. 1. MGLP circuit example, where $f = (x_1 \neg\vee x_2) \wedge [(x_3 \neg\vee 1) \vee ((x_3 \neg\vee x_4) \wedge (x_4 \neg\vee 1))] \vee (x_3 \neg\vee x_5)$

A logic path is ON *iff* all its series connected PXGs are ON simultaneously. The PXGs can be seen as logic primitives whom electrical behavior resembles a CMOS transmission gate with an embedded XNOR Boolean functionality. More specifically, each PXG is electrostatically controlled through primary input logic signals that tune the equivalent resistance of the PXG itself; a PXG fed by logic signals having same polarity shows a low-resistance, the ON state, whereas logic signals with opposite polarity lead the PXG to high-impedance, the OFF state.

As a further step to achieve a higher level generalization of the Pass-XNOR Logic (PXL) style, [25] introduces the concept of *Multi-Gate Pass Logic* (MGPL). The physical primitives of an MGPL network are generic *pass-gates* (PGs), that, from a functional point of view, can be seen as *function*-controlled switches. Similar to PXGs, they consist of two *logic-terminals* fed by the input logic signals (x and y in Figure 1-(b)), and two *transmission terminals*, one playing as the source of an evaluation signal and the other as the collector (S and D in Figure 1-(b)). The control function is a two-input Boolean operator $g(x, y)$ between the x and y logic inputs; when $g(x, y) = 1$ the PG is ON (low-resistance), Figure 1-(b), when $g(x, y) = 0$ the PG is turned OFF (high-impedance), Figure 1-(c). PGs with different control functions can be designed depending on the technology in use; the example in Figure 1 is for a NAND-PG, i.e., $g(x, y) = x \neg\vee y$. Notice that an MGPL circuit can contain PGs with different embedded functions. Similar to PXL, an MGPL (Figure 1-(d)) consists of logic paths connected in parallel between a clocked-power supply (the root) and the main output (the leaf). Each path consists of a cascade of independent PGs driven by primary inputs. When activated (all PGs turned-ON), a logic path creates a low-resistive gateway through which the clocked-power signal can flow from the root to the leaf. Under this condition the circuit's output is evaluated as 1-logic. Logic paths are in mutual exclusion by construction, that is, for a given input pattern one and only one path can be eventually activated. When there are no activated

paths the circuit's output is evaluated as 0-logic. An MGPL circuit can be modeled using a new dedicated abstract model, the Multi-Function Pass-Diagram (MFPD), Figure 1-(a), described in the next section.

As for other dynamic logic families, the logical computation of MGPL circuits consists of two distinct phases: the *configuration* phase and the *evaluation* phase. During the former, primary logic inputs, i.e., the literals composing the logic function, are fed to the logic inputs of the pass-gates. At the end of this phase the doping profile of each and every device is fixed and the resistive paths of the network are set up. In the evaluation phase the clocked power signal is precharged and propagated through the network. A pulse detected on an output leaf evaluates the implemented function as TRUE; in this regard, a Sense Amplifier can be used for each output cone in order to quickly identify the 1-logic and reshape the clock-supply signal [27].

It is worth emphasizing that although the MGPL resembles the PTL structure, the difference is substantial. In PTL circuits, transistors are used as switches that deviate the current flow to different paths; on the contrary PGs are used as switches to open/close a logic path. This is reflected by the model used to represent the circuit. Indeed, BDDs are not the most intuitive representation as PG gates do not implement any deviation of the signal. Second, while in PTL an output is always connected to a static power supply terminal, V_{dd} if '1' or Gnd if '0', output evaluation in MGPL logic is dynamic: current is flowing if '1', not flowing if '0'. Alternatively, one can see MGPL circuits as a half way between CMOS and PTL. As in CMOS series/parallel connections between gates are available, as in PTL, information is carried out by means of root-to-leaf current flow.

2.1 Pass-Gate Devices

New logic primitives introduced by emerging technologies represent a perfect fit to the structure of PGs. Figure 2 pictorially describes some of them. In particular, Figure 2-(a) shows four PG embodiments using Ambipolar Silicon-NanoWires (SiNW) [28]. The first two (left) are composed of a single SiNW transistor and implement the AND and NOR logic gates. The remaining two (center and right) consist of a pair Si-NW transistors and implement the XNOR or XOR logic gates.

Figure 2-(b) shows two possible pass-gates using standard MOSFET transmission-gates. The first one (left) implements the AND, whereas the second one (right) implements the NOR. Since both configurations require four MOSFETs, silicon devices have less expressive power when compared to SiNWs.

Finally, Figure 2-(c) shows pass-gates mapped on graphene p-n junctions [29]. A graphene p-n junction consists of two metal back-gates (blue and green triangles) driven by logic signals (x and y). Logic signals with same polarity turn the junction ON. The first PG (top left) implements the NOR gate; the outer input connections x and y are both compared to a logical-0 reference. It works as follows: when both x and y are set to 0-logic, the input evaluation signal (red

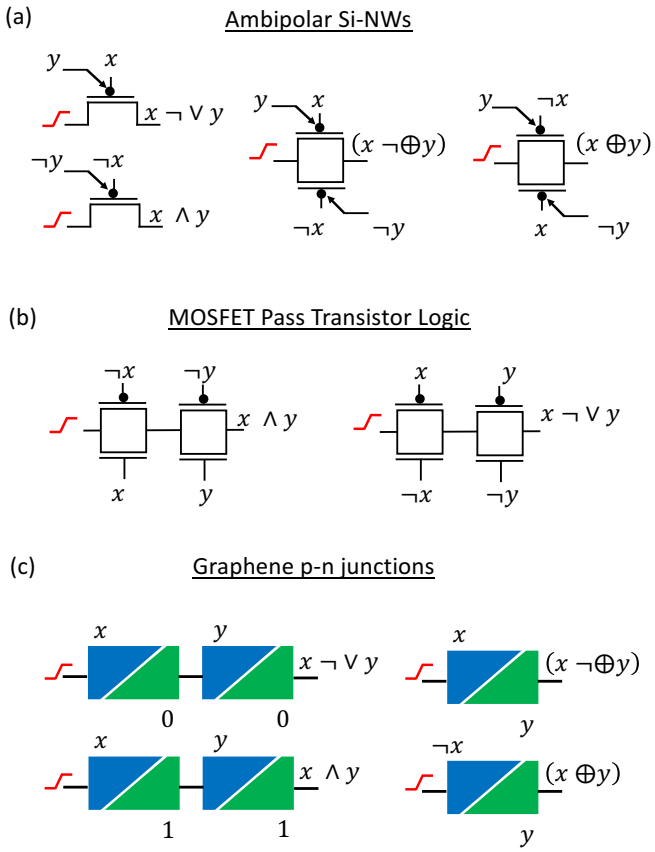

Fig. 2. Possible PGs for different logic primitives.

ramp) is allowed to propagate; in all the remaining cases at least one p-n junction is OFF and the evaluation signal is stopped. Similarly, the second pass-gate (bottom left) implements the AND; the evaluation signal propagates *iff* both x and y are fed wit 1-logic. Notice that for NOR and NAND SiNWs need less devices (1 vs. 2). The last two pass-gates (top and bottom right) implement the the XNOR and XOR gates. In this case graphene shows higher expressive power than SiNW. It is therefore clear how different technologies can be better exploited using different logic primitives.

2.2 Delay and Power Modeling of MGPL circuits

The total delay D_p of an MGLP logic circuit can be estimated as the sum of delays due to the configuration phase D_{conf} and the evaluation phase D_{eval}, as described in (1), where D_{conf} is the time primary logic inputs take to charge

$$D_p = D_{conf} + D_{eval} \tag{1}$$

the parasitic capacitances at the back-gates, whereas D_{eval} is the propagation delay of the input pulse through the front resistive paths of the network.

The amount of power consumed during the configuration phase P_c is due to charging/discharging of the input gate capacitance of the PGs. For a circuit made up N gates, an approximate, yet accurate model borrowed from CMOS is reported in (2), with P_{PG_i} as the power consumed from the i-th PG, V_{dd} as the supply voltage, f the operating frequency, C_i the input capacitance of the i-th PG, and $E_{SW_{i,j}}$ representing the probability that the input signal makes a transition.

$$P_c = \sum_{i}^{N} P_{PG_i} = \sum_{i}^{N} \sum_{j=1}^{2} 0.5 V_{dd}^2 \cdot f \cdot C_{in} \cdot E_{sw_{i,j}} \tag{2}$$

During the evaluation phase, once primary inputs have settled and PGs have been turned-ON or OFF, the circuit simply reduces to an equivalent resistor R_{eq}, i.e., the sum of the ON resistances R_{ON} of the PGs belonging to the ON-path, in series with the output load capacitance C_l. The average power consumed P_e can be therefore obtained as described in (3), where t_{rf} is rise/fall output transition time, and $i_{C_l}(t)$ is the current charging C_l. Notice that P_e is consumed iff the output is TRUE, while it is almost zero otherwise.

$$P_e = \frac{1}{t_{rf}} \int_0^{t_{rf}} R_{eq} i_{C_l}^2(t) dt = \frac{R_{eq} C_l^2}{t_{rf}^2} V_{dd}^2 \tag{3}$$

Moreover, for values of T_{rf} large enough, the evaluation phase completes at zero-power, namely, *adiabatically*.

3 Building MFPDs

3.1 Multi-Function Decomposition

The decomposition of a logic function through the primitives made available by the technology in use represent a fundamental step of any logic synthesis algorithm. Since most techniques leverage multi-level logic representations, in this section we illustrate an *ad-hoc* decomposition that is fine-tuned for pass-gates logic. Such decomposition, named *multi-function* decomposition, relies on the basic assumption that any Boolean equation given in the form of *sum-of-products* (SOPs), or *product-of-sums* (POSs), can be decomposed by means of a user-defined set of logic connectives $\mathcal{G} = \{g : B^2 \rightarrow B\}$. Let us assume a function $f(S)$ with support-set $S = \{x_1, x_2, x_3\}$ described with the following SOP:

$$f(S) = (x_1 \wedge \neg x_4) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) \tag{4}$$

By resorting to the distributive property and the identity rule, it is possible to expand the function $f(S)$ as a sequence of cubes, having cardinality of two literals:

$$\begin{aligned} f(S) = &(x_1 \wedge \neg x_4) \vee (\neg x_1 \wedge \neg x_2) \\ &\wedge (x_3 \wedge 1) \vee (x_1 \wedge x_2) \wedge (x_3 \wedge 1) \end{aligned} \tag{5}$$

Each product can then be rewritten using the Boolean connectives $g \in \mathcal{G}$ by means of duality. For instance, let us assume the availability of two connectives $\mathcal{G} = \{\{x \neg\vee \ y\}, \{x \neg\oplus \ y\}\}$, where the first one, the NOR ($\neg\vee$ symbol), has higher priority, i.e., is processed first. This means that the function $f(S)$ could be *NOR*-decomposed as shown in (6).

$$\begin{aligned} f(S) = &(\neg x_1 \neg\vee x_4) \vee (x_1 \neg\vee x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \\ &\vee (\neg x_1 \neg\vee \neg x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \end{aligned} \tag{6}$$

Such reformulation reveals that $(\neg x_3 \neg\vee \neg x_3)$ is a common term, that can be factorized as reported in (7).

$$\begin{aligned} f(S) = &(\neg x_1 \neg\vee \neg x_4) \\ &\vee (\neg x_3 \neg\vee \neg x_3) \wedge [(x_1 \neg\vee x_2) \vee (\neg x_1 \neg\vee \neg x_2)] \end{aligned} \tag{7}$$

At this point, the second operator in $\mathcal{G}$, the XNOR ($\neg\oplus$ symbol), could be applied on (7) in order to further reduce the number of literals in (5). Indeed, the term $(x_1 \neg\vee x_2) \vee (\neg x_1 \neg\vee \neg x_2)$ can be represented as the XNOR between x_1 and x_2. We refer to this operation as *Boolean substitution*. Eventually, the final result of the multi-function decomposition, the original function (4) is decomposed as described in (8).

$$f(S) = (\neg x_1 \neg\vee x_4) \vee (x_1 \neg\oplus x_2) \wedge (\neg x_3 \neg\vee \neg x_3) \tag{8}$$

Similarly, it is possible to assume a library of logic connectives described as $\mathcal{G} = \{\{x \neg\wedge y\}, \{x \neg\oplus y\}\}$, where the symbol $\neg\wedge$ denotes the NAND operator. In this case, the same Boolean function described in (4) is *NAND*-decomposed as described in (9), and thus optimized by means of the *XNOR* connective, as reported in (10).

$$\begin{aligned} f(S) = &(\neg x_1 \neg\wedge \neg x_4) \vee \neg(\neg x_1 \neg\wedge \neg x_2) \\ &\wedge (\neg x_3 \neg\wedge x_3) \vee (\neg x_1 \neg\wedge x_2) \wedge (\neg x_3 \neg\wedge x_3) \end{aligned} \tag{9}$$

$$f(S) = (\neg x_1 \neg\wedge \neg x_4) \vee (x_1 \neg\oplus x_2) \wedge (\neg x_3 \neg\wedge x_3) \tag{10}$$

It is easy to check the Boolean equivalence between (8), (10) and the original function (4); in terms of savings, both (8) and (10) show 25% literal savings.

As far as the efficiency is concerned, the proposed multi-function decomposition is closely related to (i) the set of Boolean operators and (ii) their priority ordering in $\mathcal{G}$. Although several options do exist, we resort to a technology-instructed strategy, namely, available operators in $\mathcal{G}$ are sorted, from highest to lowest, in terms of their *expressive power* (EP), which describes the ratio between the complexity of the logic operator and the number of devices needed to implement the corresponding logic gate. This guarantees the high flexibility and orthogonality of our tool onto different technologies.

As will be shown later in the text, different primitives are used during different stages of the multi-function decomposition. For the sake of clarity we define the first operator in $\mathcal{G}$, the one with the highest EP, as the *primary* primitive, the remaining ones as the *secondary* primitives.

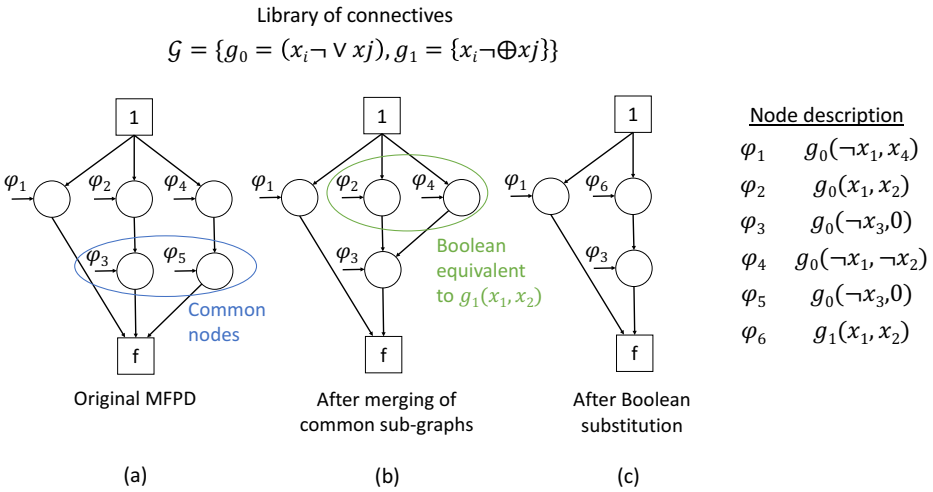

Fig. 3. MFPD of function of Equation (6) before optimization (a), after merging of common sub-graphs (b), and after Boolean substitution (c).

3.2 Multi-Function Pass Diagrams (MFPDs)

The synthesis of MGPL circuits needs an abstract model for reasoning and optimization. We introduce the MFPD, a simple, yet efficient abstract model for one-pass synthesis of MGPL circuits.

Given a generic multi-input/single-output Boolean function f with support-set $S = \{x_1, ..., x_N\}$, its MFPD (Figure 3) representation is a polarized, directed acyclic graph defined as $G = (\Phi \cup V \cup \Theta \cup A)$. The set of internal nodes $v \in V$ are labeled as $g(x, y)$, with $g \in \mathcal{G}$ a two-input primitive Boolean connective and $x, y \in S$. Each internal node v has one outgoing edge $a \in A$ representing the logical conjunction (AND) with the successor node. The terminal node with *indegree* 0 represents the root of the MFPD, where the function starts to be evaluated; the terminal node with *outdegree* 0 is the leaf of the MFPD, the output of the function f. Multiple output functions are represented by many MFPDs as the number of outputs. As an example, Figure 3-(a) shows the MFPD structure for the function (6) with $g_0 = (x_i \neg \vee x_j)$ and $g_1 = (x_i \neg \oplus x_j)$.

The main strengths of an MFPD are three. First, they guarantee the capability of supporting multi-function decomposition. This degree of freedom comes at the cost of canonicity, that is, MFPDs do not have an unique representation of Boolean formulae. However, relaxing the canonicity constraint is a well-accepted concept in the EDA community; indeed, And-Inverter Graphs (AIGs) integrated into commercial multi-level logic synthesis tools are non-canonical representations, but nonetheless they are likely used because more compact and manageable. Second, a MFPD has a 1-to-1 mapping to the final circuit implementation, that is, each internal node is implemented by a single logic primitive in the resulting network (please refer to Figure1-(a)). This makes the MFPD a

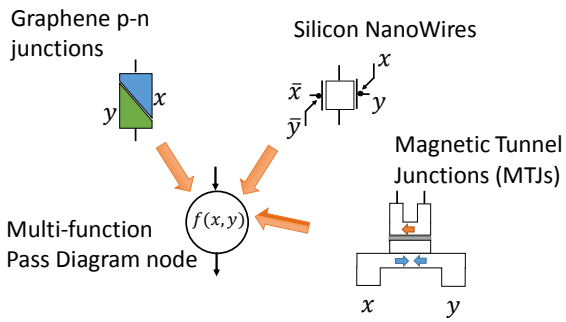

Fig. 4. Multi-Function Pass Diagram node mapping with different technologies.

universal model for both the optimization stage and the technology mapping, a key aspect for *one-pass logic synthesis* [22]. Third, but not for importance, MFPDs may serve as model for MGPL circuits mapped using different emerging devices. As an example, Figure 4 depicts a generic MFPD internal node, with $f = x \neg\oplus y$, and its implementation using Graphene p-n junctions, Silicon-NanoWires, Magnetic Tunnel Junctions and Memristors. Thus, the proposed MFPD model and its synthesis methodology can be seen as orthogonal tools for the assessment of a wide range of emerging technologies.

4 Algorithms

4.1 Building MFPD

Algorithm 1 describes the pseudo-code of the *Build* routine we implemented for multi-function decomposition and MFPD construction.

The main input parameters are (i) a tabular description T of the Boolean function and (ii) the primary connective (the first operator in the list of primitives $\mathcal{G}$). Table T can be a non-minimized *implicant* table (i.e., not prime) and can be obtained through any Verilog compiler, e.g., *ABC* [30]. We refer to T as the *PLA* table. As an example, Table 1 shows the PLA table for the Boolean function (5), where the character '-' identifies a dont'care.

Table 1. PLA table of function (5)

$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	x_4	**f**
1	-	-	0	1
0	0	1	-	1
1	1	1	-	1

Algorithm 1: MFPD build

Input: PLA Table T, Primary connective $g_0 \in \mathcal{G}$
Output: Multi-Function PD $MFPD$

```
1  MFPD = ∅
2  foreach row R ∈ T do
3      CUBES_R = ∅
4      DontCareSet = DetermineDCS(R)
5      foreach primary input PI ∈ R do
6          if PI ∉ DontCareSet then
7              CUBES_R.append(PI)
8          end
9      end
10     foreach v_{i,k} ∈ CUBES_R do
11         NewNode ← SetPolarity(v_{i,k}, g)
12         PT_R.append(NewNode)
13     end
14     MFPD.append(PT_R)
15 end
```

The MFPD is generated branch-wise, that is, for each row of the PLA table, i.e., for each product term of the function, nodes are appended in series by iterating the following sequence of operations:

Cube sequence generation (line 3-9) – variables not belonging to the don't-care set are included in the cube list $CUBES_R$ in order of appearance; those belonging to the dont-care set are dropped. For odd sequences, the last variable is paired with '1' logic so as to maintain Boolean equivalence. For instance, considering Table 1, for the first row $CUBES_1 = \{(x_1, \neg x_4)\}$, for the second row $CUBES_2 = \{(\neg x_1, \neg x_2), (x_3, 1)\}$, for the third row $CUBES_3 = \{(x_1, x_2), (x_3, 1)\}$.

Node generation (line 10-14) – for each pair of cubes stored in $CUBES_R$, the polarity of the variables are fixed according to the *primary* Boolean connective g and the resulting nodes are appended on the current branch. Let us consider $CUBES_2$ which contains two cubes, $(\neg x_1, \neg x_2)$ and $(x_3, 1)$; with g the NOR operator (like the example in Section 3.1), variables are complemented (by De-Morgan) as (x_1, x_2) and $(\neg x_3, \neg x_3)$ respectively.

Given a table T with N implicants and M literals, the proposed *build* routine has a complexity of $O(N \cdot M)$.

4.2 Optimization

Algorithm 2 describes the pseudo-code of the optimization stage for redundancy removal. It implements two different optimization techniques: *(i)* node elimination by *Boolean substitution*; *(ii)* merging of isomorphic sub-graphs. While the latter is reminiscent of standard reduction rules from BDDs [23], the former one

Algorithm 2: MFPD optimization algorithm

```
   Input: MFPD, Secondary connectives G = (g_1, ... g_m) ∈ 𝒢
   Output: Optimized Multi-Function PD OMFPD
 1 OMFPD = ∅
 2 foreach path P ∈ MFPD do
 3     C_M ← ∅
 4     C_E ← ∅
 5     foreach path Q ∈ MFPD, with Q ≠ P do
 6         if SameSupport(P, Q) then
 7             if CheckBoolSub(P, Q, G) then
 8                 C_E.append(Q, g_k ∈ G)
 9             end
10         else
11             if SharedNodes(P, Q) then
12                 C_M.append(Q)
13             end
14         end
15     end
16     if |C_E| > 0 then
17         M ← ApplyBoolSub(P, C_E, G)
18     else if |C_M| > 0 then
19         M ← MergeIsomorphic(P, C_M)
20     OMFPD.append(M)
21 end
```

is an ad-hoc strategy for MFPDs. Its purpose is to find suitable equivalent logic connectives, among the list of *secondary* connectives in $\mathcal{G}$, that can be eventually substituted in order to enable node elimination and reduce the MFPD cardinality; as illustrated in the examples of Section 3.1. Please note that *secondary connectives* are selected with a greedy approach, that is, the first one that satisfies the Boolean equivalence is instantiated in the network.

Input parameters of Algorithm 2 are the MFPD obtained through the *MFPD Build* routine, and the list of secondary connectives $G \in \mathcal{G}$.

Candidate selection (line 3-15) – Each root-to-leaf path P of the MFPD is compared with any other path Q ($P \neq Q$). If (line 6) P and Q share the same support set (i.e., nodes in P and Q are driven by the same literals) the algorithm checks (line 7) whether it is possible to perform a *Boolean substitution*, namely, it checks whether some of the operators associated with the nodes in Q can be substituted with some other operator $g_k \in G$ s.t. Boolean equivalence is satisfied. If so, P and Q share a common node expressed by means of g_k, that allows to merge P and Q in a single path. Therefore, Q is stored in the candidates list C_E together with the connective g_k that enables its elimination. If P and Q do not have common support set (line 10), the algorithm checks whether a path

Q shares at least one node with P; if so, Q is a potential candidate for node merging and it is temporarily stored in the list of candidates C_M.

Merge and Eliminate (line 16-20) – once candidates have been selected, the algorithm first evaluates whether there exists at least one candidate for node elimination by *Boolean substitution* ($|C_E| > 0$). If so, the common node between P and C_E is replaced with the new connective g_k, and redundant paths in C_E are removed (*ApplyBoolSub* function). If not and the list C_M is not empty, then common nodes between C_M and P are evaluated for merging through the *MergeIsomorphic* function.

Figure 3-(b) and 3-(c) show the results of the optimization procedures described above applied on the MFPD obtained through the build function (Figure 3-(a)).

Since Boolean elimination guarantees higher benefits in terms of node count and circuit complexity, e.g., it reduces the number of common branches due to merge operations, this characteristic is exploited first during the last phase of the optimization process with the $MergeAndRemap$ function. Otherwise, common nodes between the $CANDIDATES_M$ list and P are merged by means of the $MergeMaxSharing$ function. The final result is then appended in the optimized MFPD structure $OMFPD$. Applying this procedure to the MFPD depicted in Figure 3-a allows us to achieve a more compact representation of the same function by means of only three logical nodes/operations, as depicted in Figure 3-b.

Concerning complexity, since each path P is compared with any other path Q, the total number of loops is $\frac{N \cdot (N-1)}{2}$, with N the number of paths in the starting MFPD. The complexity of the optimization routine is $O(N^2)$. Notice that all other sub-routines have a $O(1)$ complexity (operations are completed in constant time) except for functions $SameSupport$ and $SharedNodes$ which show a complexity of $O(M)$, with M the number of nodes in the path Q.

5 Simulation Results

Experimental results reported in this section provide a fair comparison of MFPDs and the resulting MGPL circuits mapped onto different technologies against state-of-the-art data-structures and PTL circuits. The goal is to demonstrate that:

1. MFPDs optimized with the eduction rules described in Section 3 give substantial savings.
2. MFPDs represent the most compact solution for the representation of switching functions implemented through the MGPL style; metrics adopted are *expressive power*, namely the number of nodes and *depth*, i.e., nodes count along the longest path.
3. MFPDs are a true technology-independent abstract model, namely, the way they model a Boolean function can be orthogonally applied to any kind of technology and, most importantly, whatever the logic primitives are.

4. MGPL circuits obtained through MFPDs might become the vehicle for ultra-low power computing using emerging technologies, graphene in particular; in this regard, we show that the MGPL style allows large gains w.r.t. PTL and, most importantly, it is well suited for ultra-low power digital circuits.

We set up five different synthesis flows, the first four are for pass-gate logic circuits, the target of this work, the fifth one is for standard cells-based circuits.

- **MFPDs** (the solution proposed in this work): circuits described using the PLA format [30] are processed with our tool *Kanon* for multi-function decomposition using the connective set $\mathcal{G} = \{\{x\neg \vee y\}, \{x\neg\oplus y\}, \{x \oplus y\}\}$. It is worth to notice that even though more Boolean operators can be used, here we force our tool working in worst-case conditions where only three primitives are allowed. Resulting MFPDs are then mapped onto MGPL circuits using different technologies.

- **PDs** (introduced in [24]): circuits described using the PLA format [30] are processed leveraging *Gemini*, a single-function XNOR decomposition tool; resulting PDs are mapped onto PXL circuits using different technologies.

- **Biconditional-BDDs** (described in [21]): circuits are first synthesized using a standard multi-level synthesis tool and then translated into BDDDs using single-function XOR decomposition scheme; the resulting BBDDs are mapped onto a tree-based PTL-like structure using different technologies.

- **BDDs**: circuits are processed with a C program that leverage the CUDD package [23]; BDD structures, obtained with a single-function MUX-based decomposition, are mapped on PTL-like circuit using different technologies.

- **AIGs**: obtained with the ABC synthesis tool [30], the AIGs are mapped on a CMOS library containing only AND and INV gates; it is worth emphasizing that AIGs cannot be directly used for pass-gates logic circuit, however, they serve as a reference point to better evaluate MFPDs.

The experiments were run on a set of open-source benchmarks from the LGSynth91 suite [31], and accurate SPICE simulations were used for the characterization of the obtained netlists. Please note that the size of such benchmarks is comparable to that of those used in other synthesis-related works, e.g., [21]. Without loss of generality, only combinational logic cones have been considered for synthesis, i.e., in-to-out and register-to-register logic cones.

5.1 Efficiency of reduction rules during MFPD optimization

Table 2 gives a brief description of the adopted benchmarks and an overview of the efficiency of the algorithms proposed in this work. Columns **PI**, **PO** and **I** represent the total number of primary inputs, primary outputs and implicants of each benchmark. Under the label **Number of nodes**, column **w/o opt** refers

Table 2. MFPD reduction rule efficiency.

	PI	PO	I	Number of nodes		
				w/o opt	w/ opt	Savings %
sao2	10	4	58	229	152	33.62
o64	130	1	65	65	65	-
5xp1	7	10	75	161	111	31.06
c8	28	18	79	156	108	30.77
duke2	22	29	87	401	287	28.43
apex1	45	45	206	921	677	26.49
misex1	8	7	32	67	31	53.73
misex2	25	18	29	101	75	25.74
b12	15	9	431	1007	579	42.50
k2	45	45	936	3791	2103	44.53
bigkey	486	421	6151	19054	10771	43.47
s13207.1	700	790	10987	53868	33005	38.73
			Total	**79821**	**47964**	**Avg. 39.91**

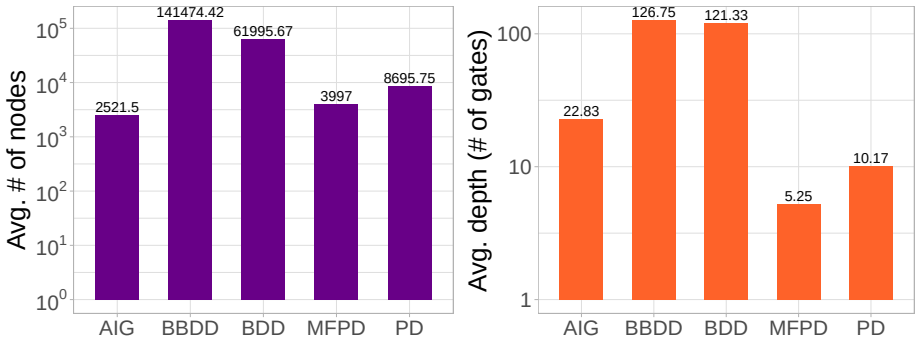

Fig. 5. Binary MFPDs efficiency w.r.t. PDs, BDDs, BBDDs and AIGs. Average number of nodes (left), average depth (right).

to MFPDs after the build process, whereas column **w/ opt** refers to MFPDs after optimization; column **Savings** reports optimization gain;

As the table suggests, the proposed reduction rules allow about 40% of savings on average. Noticeably, large savings have been recorded for all the benchmarks, except for `o64`. For this case we observed the PLA table is a diagonal matrix of '1s' which prevents MFPD optimization.

5.2 Compactness of the MFPD model and execution time

Figure 5 depicts the obtained synthesis results averaged over all the benchmarks described in Table 2. The plot suggests that MFPDs outperform BBDDs, which are 35.39x larger, BDDs (15.51x larger) and PDs (2.1x larger); only AIGs are more compact (0.63x). Indeed, AIGs leverage two key options available in multi-level optimization, namely the possibility of reusing cascades of common sub-

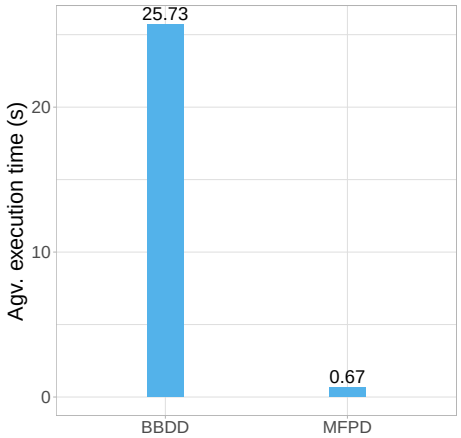

Fig. 6. Average device count after synthesis and mapping. Graphene (left), Ambipolar Si-NWs (center), MOSFET PTL (right).

expressions and local don't-care conditions, a technique that is not applicable to pass-gates logic styles.

However, an important aspect concerns the depth of the data-structures. In this case MFPDs are the most efficient solution since BBDDs (21.12x deeper), BDDs (20.33x deeper) and PDs (1.83x deeper) show an higher number of devices on the longest path. AIGs do the same as they return circuit topologies 3.83x deeper. Therefore, MFPDs are well suited for pass-gates logic circuits, where smaller depth translates into shorter delays and smaller voltage noise.

Such huge savings achieved are the consequence of the efficient multi-function decomposition, in particular: (*i*) the availability of more Boolean operators w.r.t. BDDs, BBDDs and PDs, (*ii*) the fact that inputs variables belonging to the dont-care set are dropped during decomposition (please refer to Algorithm 1), (*iii*) the regularity of the implication table that allows large minimization (see Algorithm 2).

From a complexity viewpoint, the barchart in Fig. 6 reports the average CPU execution time of the *Kanon* tool compared to the BBDD package. It turns out that MFPD synthesis is, on the average, 38x faster w.r.t. the procedures used for BBDDs. As a representative example, for the largest benchmark (`s13207.1`) the MFPD is built and optimized in 7.08s, whereas the equivalent BBDD takes 241.9s. This is due to the lower computational workloads of MFPD algorithms which do not require the reconstruction of the network graph during optimization.

5.3 Many technologies, one synthesis methodology

To demonstrate the "orthogonality" of both the MFPD model and the MGPL style over different technologies, we mapped the benchmarks under analysis using

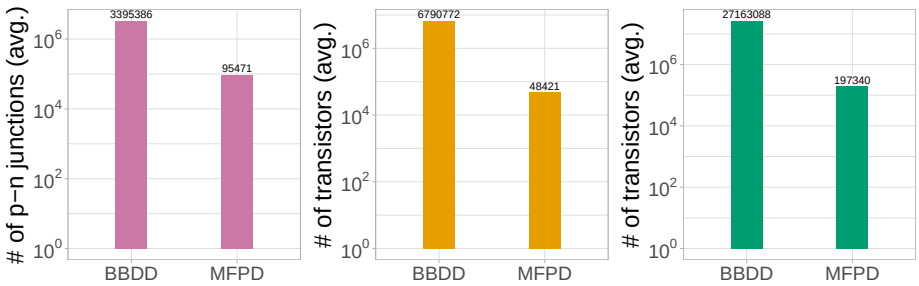

Fig. 7. Average device count after synthesis and mapping. Graphene (left), Ambipolar Si-NWs (center), MOSFET PTL (right).

three different devices: graphene p-n junctions (*Graphene*), Ambipolar Silicon NanoWires (Si-NWs) Pass-Transistors (*Si-NW PT*), and traditional MOSFET-based Pass-Transistors (*Si-MOS PT*). As described in Section 2.1, each of these technologies has different "optimal" primitives, namely the ones with the highest expressive power. Hence, tools that can seamlessly use different logic primitives may represent a genuine solution for the evaluation and comparison of different emerging technologies.

Figure 7 summarizes the post-synthesis results obtained using our tool. Since MFPDs are a superclass of Pass Diagrams, we only provide comparison against BBDD-based synthesis. BBDDs represent the most recent solution proposed for emerging technologies [21] and their superiority w.r.t. other solutions have been already demonstrated. Notice that MFPDs nodes can be mapped with NOR, XOR and XNOR, while BBDDs only allow XOR mapping. Each of this pass-gates count different devices according to the adopted technology (Fig. 7). As a result of the multi-function decomposition, circuits synthesized using MFPDs are several orders of magnitude smaller in size; this translates into more area and more power efficient MGPL circuits.

5.4 Power and Performance efficiency of MGPL circuits

The extremely compact structure of MGPL circuits allows very high power/energy reduction. While this is an intuitive conclusion, here we underline the energy efficiency of the MGPL style for emerging technologies, Graphene in particular. Figure 8 provides a technological comparison between Graphene-based MGPL circuits and Silicon-MOS PTL circuits. The plot shows the power-delay product (PDP) averaged over all the benchmarks as function of the transition time T_r of the input signals. The plot highlights the "adiabatic" nature of both implementations, i.e., PDP reduces as T_r increases. For a T_r that ranges from 1 to 1000 ps (3 orders of magnitude), the PDP of graphene reduces by more than 5 orders of magnitude, whereas that of silicon reduces only by 3 orders of magnitude. However, and this is the most important aspect, graphene circuits

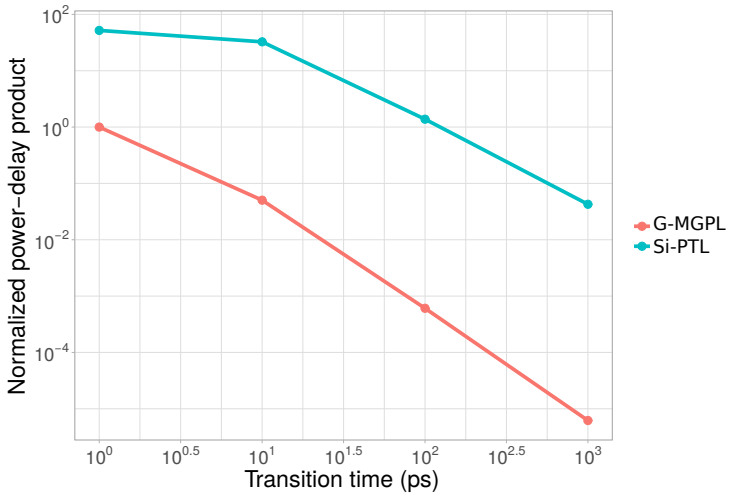

Fig. 8. Normalized PDP vs transition time.

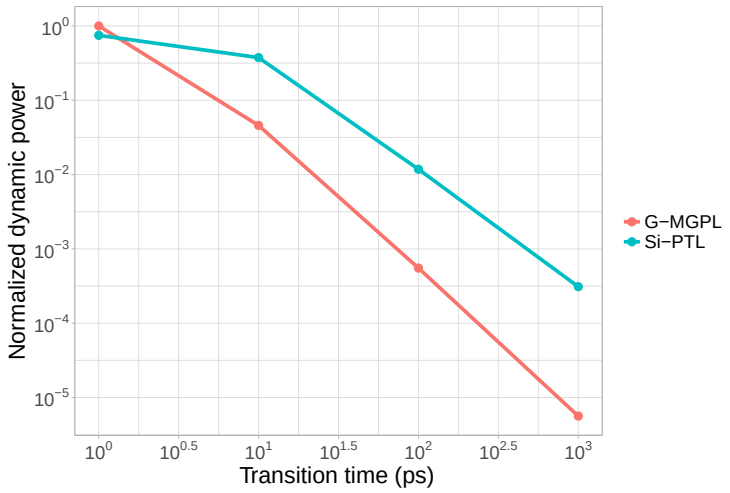

Fig. 9. Normalized dynamic power vs. transition time.

are more energy efficient, not just in terms of absolute numbers, a result due to the intrinsic characteristics of the material [24]), but also in terms of "scalability". This concept is further explained in Fig.9 that correlates dynamic power consumption over transition time. As the plot suggests, for $T_r \cong 1ps$, namely outside the adiabatic region, PTL circuits are more power efficient, but as T_r increases, the adiabatic nature of the MGPL circuits shows reaching a power consumption that is about 2 order of magnitude lower that the PTL counterpart (best case at $T_r \cong 1ns$).

6 Conclusions

In this work we introduced a novel abstract representation for Boolean switching functions: the MFPD. Such a new data-structure, obtained with a multi-function logic decomposition, allows the implementation of compact MGPL circuits that well fit the characteristics of new emerging devices.

Results obtained with our tool *Kanon* demonstrate that MGPL circuits show superior characteristics w.r.t. state-of-art solutions, in particular *(i)* larger area efficiency (almost 15.51x better than PTL circuits obtained with BDDs) and *(ii)* shallower logic paths (77% w.r.t. CMOS circuits obtained with AIG-based multi-level synthesis).

These achievements demonstrate that there is a huge margin of improvement when moving to new technologies and that solutions universally recognized as a de facto standard for CMOS may fail when considering devices with different characteristics.

References

1. M. Schulz, "The end of the road for silicon?" *Nature*, vol. 399, no. 6738, pp. 729–730, 1999.
2. S. E. Thompson and S. Parthasarathy, "Moore's law: the future of si microelectronics," *Materials Today*, vol. 9, no. 6, pp. 20–25, 2006.
3. K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance cmos variability in the 65-nm regime and beyond," *IBM journal of research and development*, vol. 50, no. 4.5, pp. 433–449, 2006.
4. R. Zimmermann and W. Fichtner, "Low-power logic styles: Cmos versus pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 7, pp. 1079–1090, 1997.
5. C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.
6. J. M. Rabaey and M. Pedram, *Low power design methodologies*. Springer Science & Business Media, 2012, vol. 336.
7. M. De Marchi, D. Sacchetto, S. Frache, J. Zhang, P. Gaillardon, Y. Leblebici, and G. De Micheli, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets," in *IEDM'12: International Electron Devices Meeting*, Dec 2012, pp. 8.4.1–8.4.4.
8. H.-Y. Chiu, V. Perebeinos, Y.-M. Lin, and P. Avouris, "Controllable pn junction formation in monolayer graphene using electrostatic substrate engineering," *Nano letters*, vol. 10, no. 11, pp. 4634–4639, 2010.
9. M. Y. Han, B. Özyilmaz, Y. Zhang, and P. Kim, "Energy band-gap engineering of graphene nanoribbons," *Physical review letters*, vol. 98, no. 20, p. 206805, 2007.
10. S. Ikeda, J. Hayakawa, Y. M. Lee, F. Matsukura, Y. Ohno, T. Hanyu, and H. Ohno, "Magnetic tunnel junctions for spintronic memories and beyond," *IEEE Transactions on Electron Devices*, vol. 54, no. 5, pp. 991–1002, May 2007.
11. S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

12. R. S. Shelar and S. S. Sapatnekar, "Bdd decomposition for delay oriented pass transistor logic synthesis," *IEEE Transactions on VLSI Systems*, vol. 13, no. 8, pp. 957–970, 2005.
13. V. G. Oklobdzija *et al.*, "Pass-transistor adiabatic logic using single power-clock supply," *IEEE Transactions on Circuits and Systems II*, vol. 44, no. 10, pp. 842–846, 1997.
14. S. Houri, G. Billiot, M. Belleville, A. Valentian, and H. Fanet, "Limits of cmos technology and interest of nems relays for adiabatic logic applications," *IEEE Transactions on Circuits and Systems I*, vol. 62, no. 6, pp. 1546–1554, 2015.
15. L. Ding, Z. Zhang, S. Liang, T. Pei, S. Wang, Y. Li, W. Zhou, J. Liu, and L.-M. Peng, "Cmos-based carbon nanotube pass-transistor logic integrated circuits," *Nature communications*, vol. 3, p. 677, 2012.
16. S. Miryala, A. Calimera, E. Macii, and M. Poncino, "Ultra low-power computation via graphene-based adiabatic logic gates," in *DSD'14: Digital System Design Conference*, 2014, pp. 365–371.
17. M. Suzuki, N. Ohkubo, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 1.5-ns 32-b cmos alu in double pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, pp. 1145–1151, Nov 1993.
18. J. D. Lee, Y. J. Yoon, K. H. Lee, and B.-G. Park, "Application of dynamic pass-transistor logic to an 8-bit multiplier," *Journal-Korean Physical Society*, vol. 38, no. 3, pp. 220–223, 2001.
19. T.-Y. Wu, L.-Y. Lu, and C.-H. Liang, "Low-leakage and low-power implementation of high-speed 65nm logic gates," in *EDSSC'08: Electron Devices and Solid-State Circuits Conference*. IEEE, 2008, pp. 1–4.
20. V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. De Micheli, "Decision diagrams and pass transistor logic synthesis," in *Int'l Workshop on Logic Synthesis*, vol. 168, 1997.
21. L. Amaru *et al.*, "Biconditional binary decision diagrams: A novel canonical logic representation form," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, no. 4, pp. 487–500, Dec 2014.
22. R. Drechsler and W. Günther, *Towards One-Pass Synthesis*. Springer Science & Business Media, 2013.
23. F. Somenzi, "Cudd: Cu decision diagram package release 2.3. 0," *University of Colorado at Boulder*, 1998.
24. V. Tenace, A. Calimera, E. Macii, and M. Poncino, "One-pass logic synthesis for graphene-based Pass-XNOR logic circuits," in *DAC'15: Design Automation Conference*. ACM, 2015, pp. 1–6.
25. V. Tenace, A. Calimera, E. Macii, and M. Poncino, "Multi-function logic synthesis of silicon and beyond-silicon ultra-low power pass-gates circuits," in *VLSI-SoC'16: International Conference on Very Large Scale Integration*, Sept 2016, pp. 1–6.
26. V. Tenace, A. Calimera, E. Macii, and M. Poncino, "Pass-XNOR logic: a new logic style for PN junction based graphene circuits," in *DATE'14: Design, Automation and Test in Europe*. IEEE, 2014, pp. 1–4.
27. V. Tenace, A. Calimera, E. Macii, and M. Poncino, "Quasi-adiabatic logic arrays for silicon and beyond-silicon energy-efficient ics," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1111–1115, 2016.
28. M. De Marchi, D. Sacchetto, S. Frache, J. Zhang, P.-E. Gaillardon, Y. Leblebici, and G. De Micheli, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets," in *IEDM'12: International Electron Devices Meeting*, 2012, pp. 4–8.

29. S. Miryala, V. Tenace, A. Calimera, E. Macii, and M. Poncino, "Ultra-low power circuits using graphene p–n junctions and adiabatic computing," *Microprocessors and Microsystems*, vol. 39, no. 8, pp. 962–972, 2015.
30. B. L. Synthesis and V. Group, "Abc: A system for sequential synthesis and verification," http://www.eecs.berkeley.edu/ alanmi/abc/, 2014.
31. "Collection of digital design benchmarks," http://goo.gl/6fOVfN, 2015.